BUY IN PRINT WORLDWIDE MAGPI.CC/STORE

The MagPi

Issue 105 | May 2021 | magpi.cc

The official Raspberry Pi magazine

Code a Pico **reaction game**

Discover amazing portable projects

RASPBERRY PI 50 HACKS & HINTS

SECRETS THAT ARE OUT OF THIS WORLD

The best of **#MonthOfMaking**

51 PAGES OF PROJECTS & TUTORIALS

WELCOME
to *The MagPi* 105

There's so much to Raspberry Pi that even we, after years of hacking, making, building and rebuilding, have something new to learn. I just learned, after all these years, that holding down **ALT** lets me move windows around in Raspberry Pi OS.

Our 50 Hacks & Hints feature (page 32) is packed with advice. Some of it will be invaluable to absolute beginners; other tips will be hidden to even seasoned experts.

Meanwhile, this year's #MonthOfMaking has come to a close, and Rob has the best of everything that our readers made during March (page 72). It's an amazing collection that showcases the ingenuity of the Raspberry Pi community.

There are a lot of video game builds to enjoy in this issue. KG's amazing arcade machine (page 42) is starting to take shape and looks incredible. And one reader has recreated the Star Wars Arcade Cabinet from 1983. Plus, if you are into coding there are some incredible tutorials this month, from making a tic-tac-toe GUI game, to building a Raspberry PI Pico reaction game.

I've loved putting together this edition of *The MagPi*.

Lucy Hattersley Editor

EDITOR

Lucy Hattersley

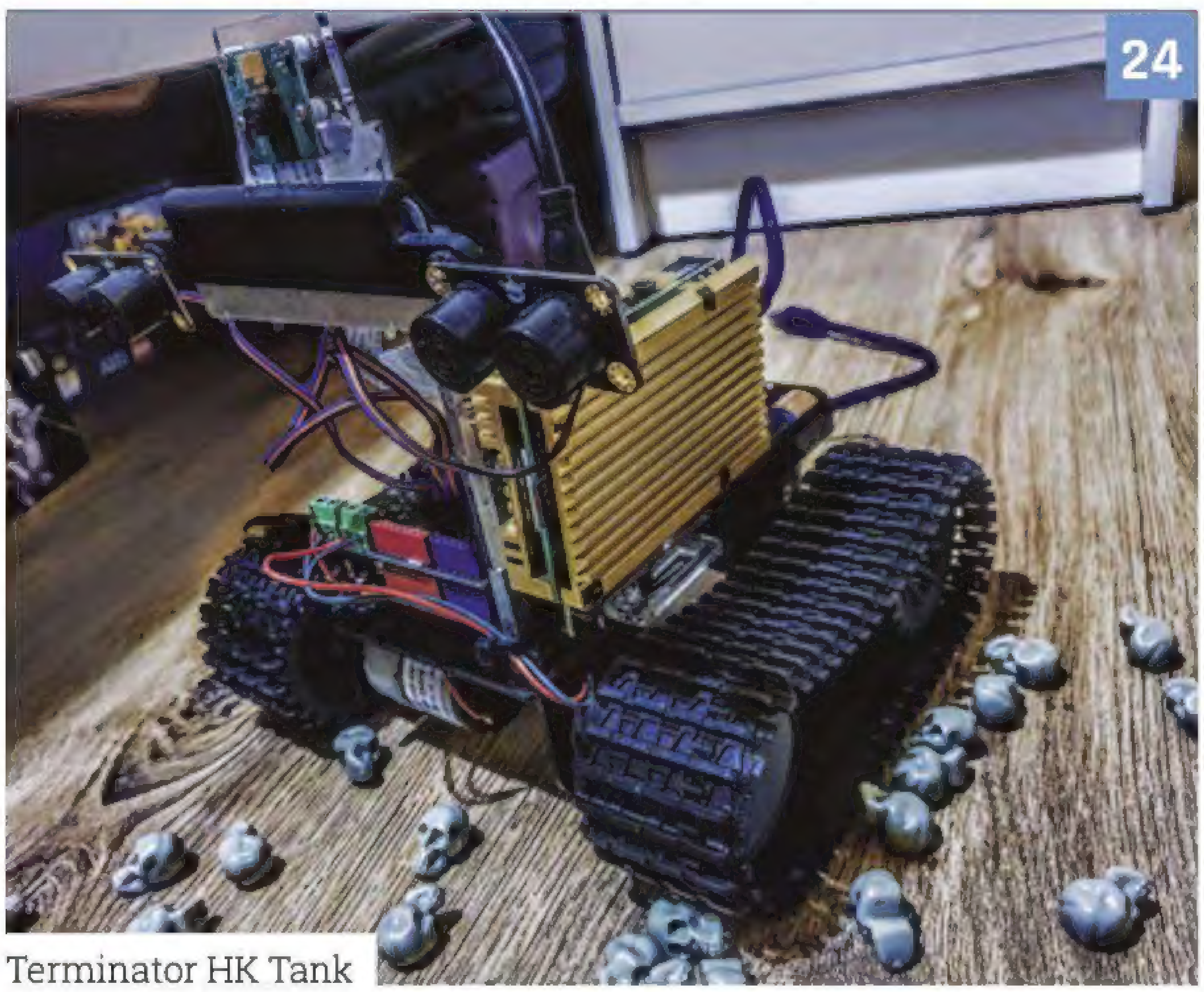
Lucy has rediscovered her love of camping this year. The trick is to buy the bounciest inflatable mattress you can find.

@LucyHattersley

Contents

> Issue 105 > May 2021

3/4 Star Wars Arcade Cabinet



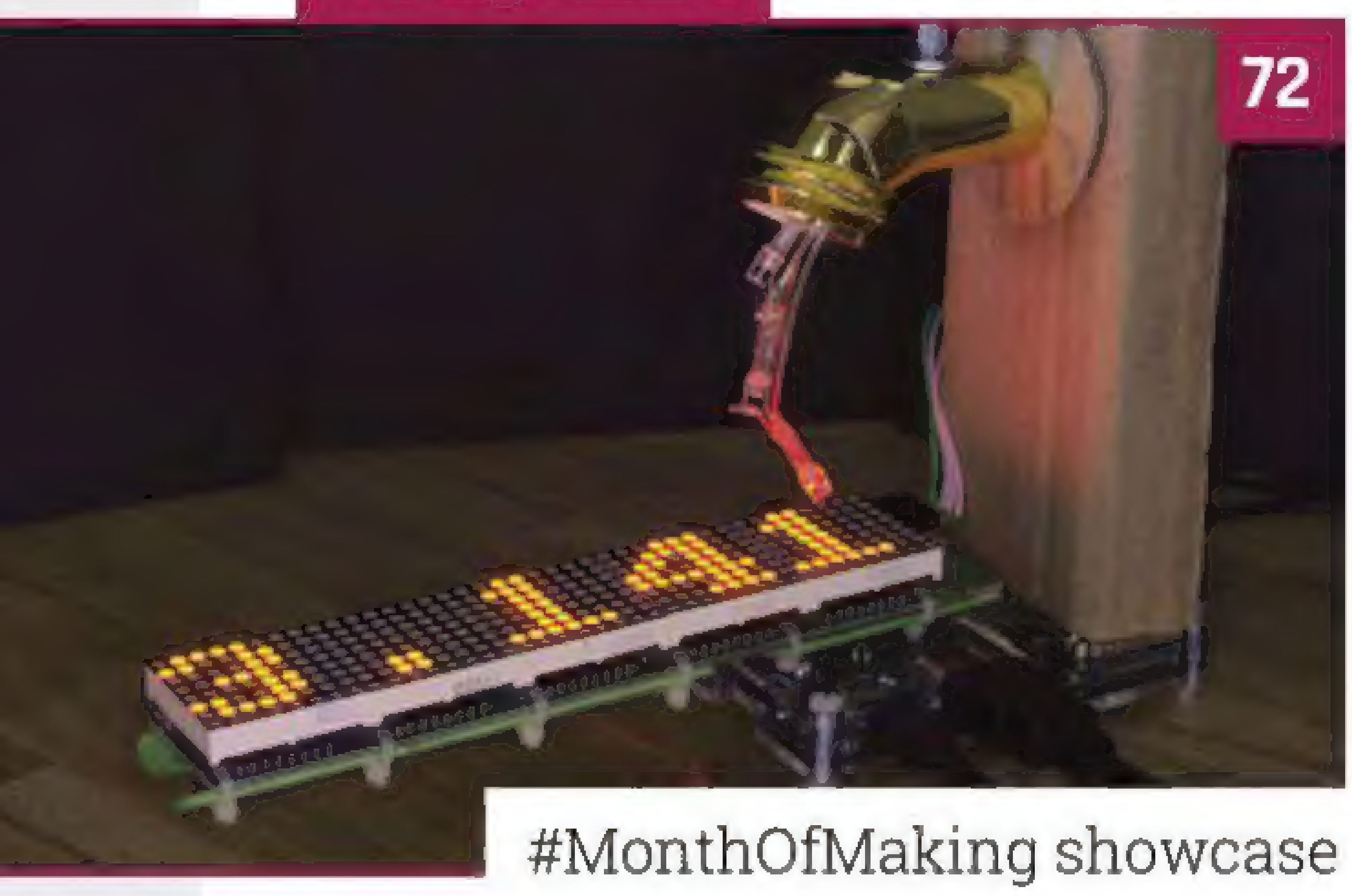
Terminator HK Tank

Tutorials

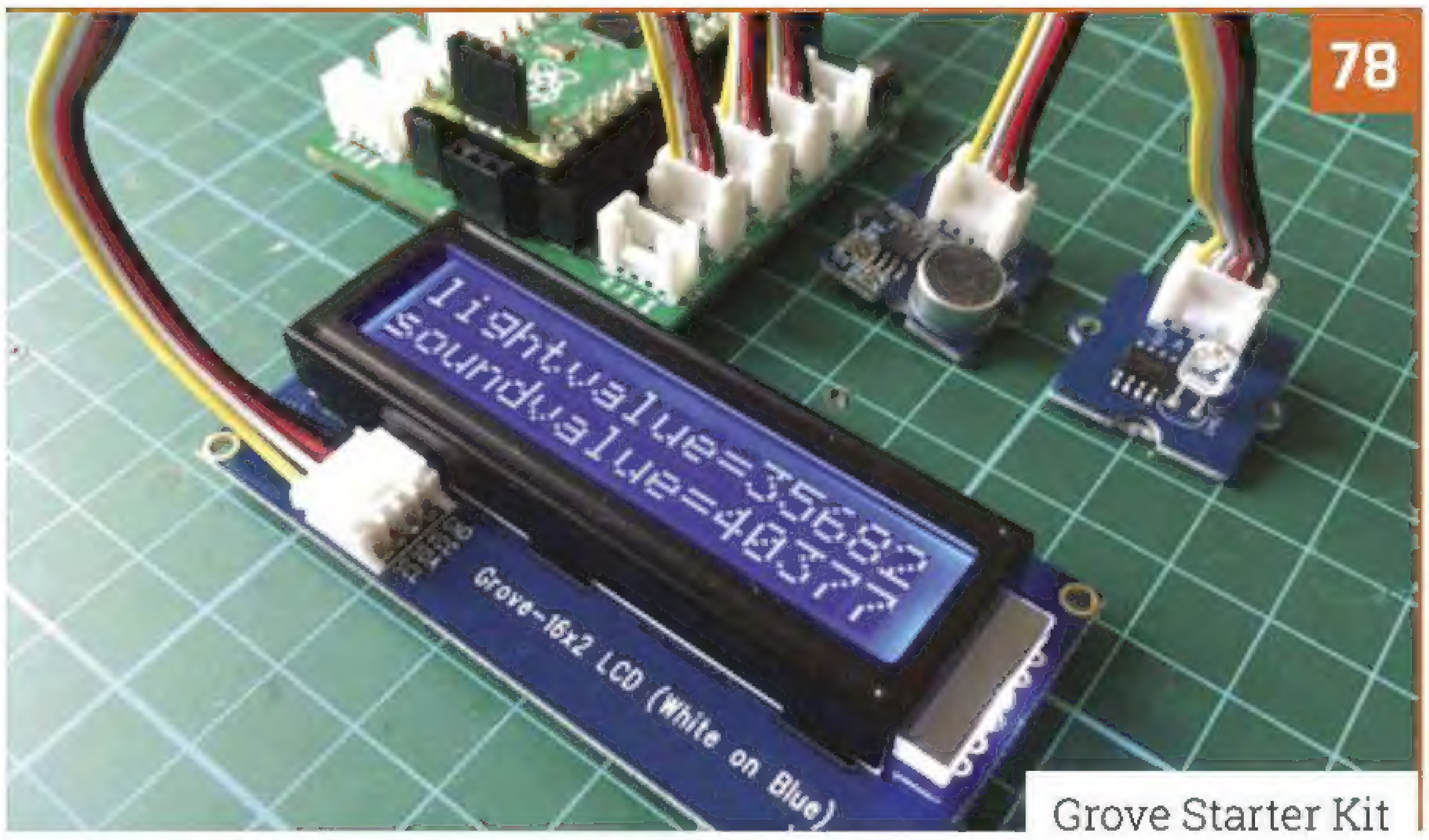
Build an arcade machine – part 2

Cheap Trills – part 4

The Big Feature



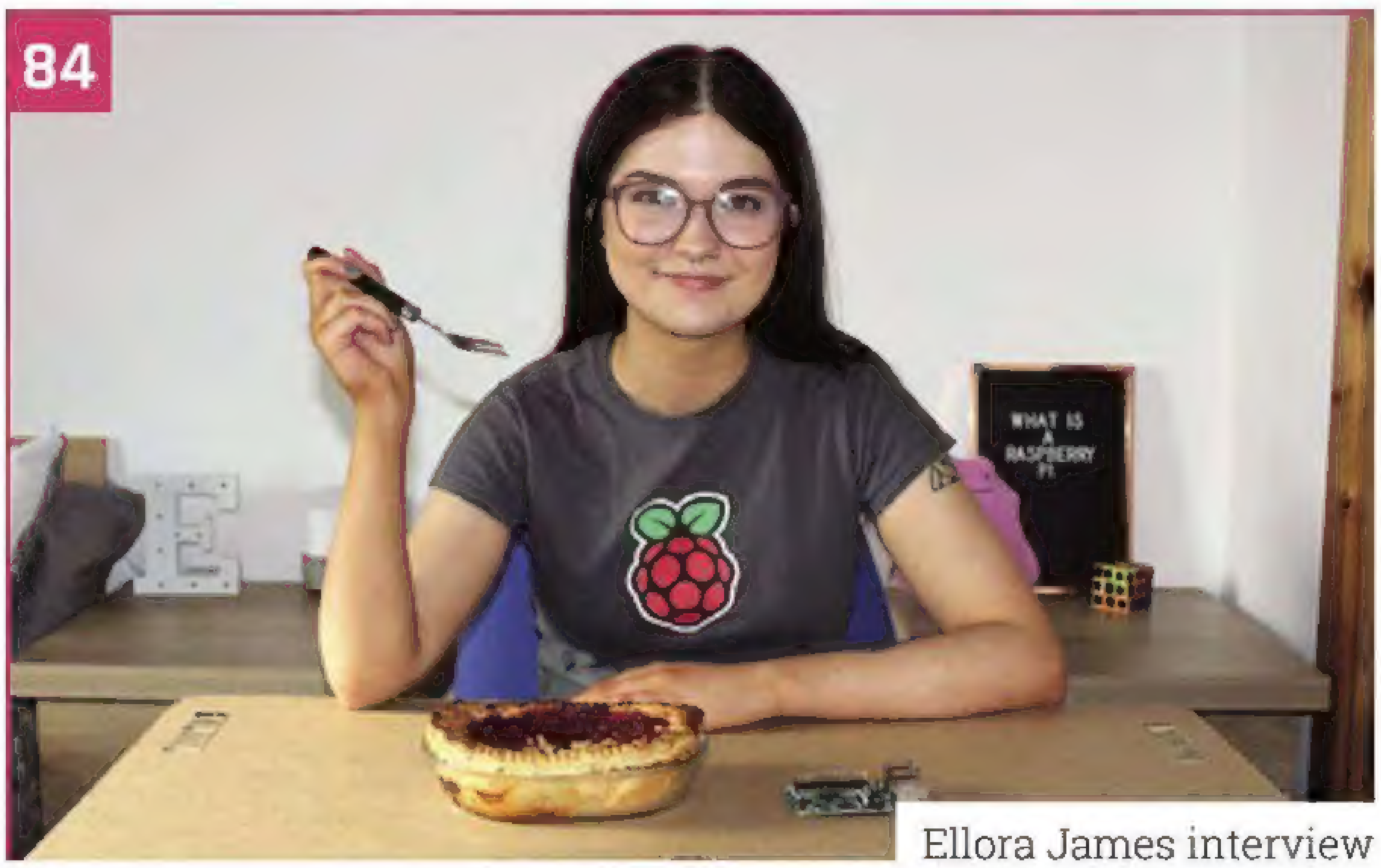
#MonthOfMaking showcase



Grove Starter Kit

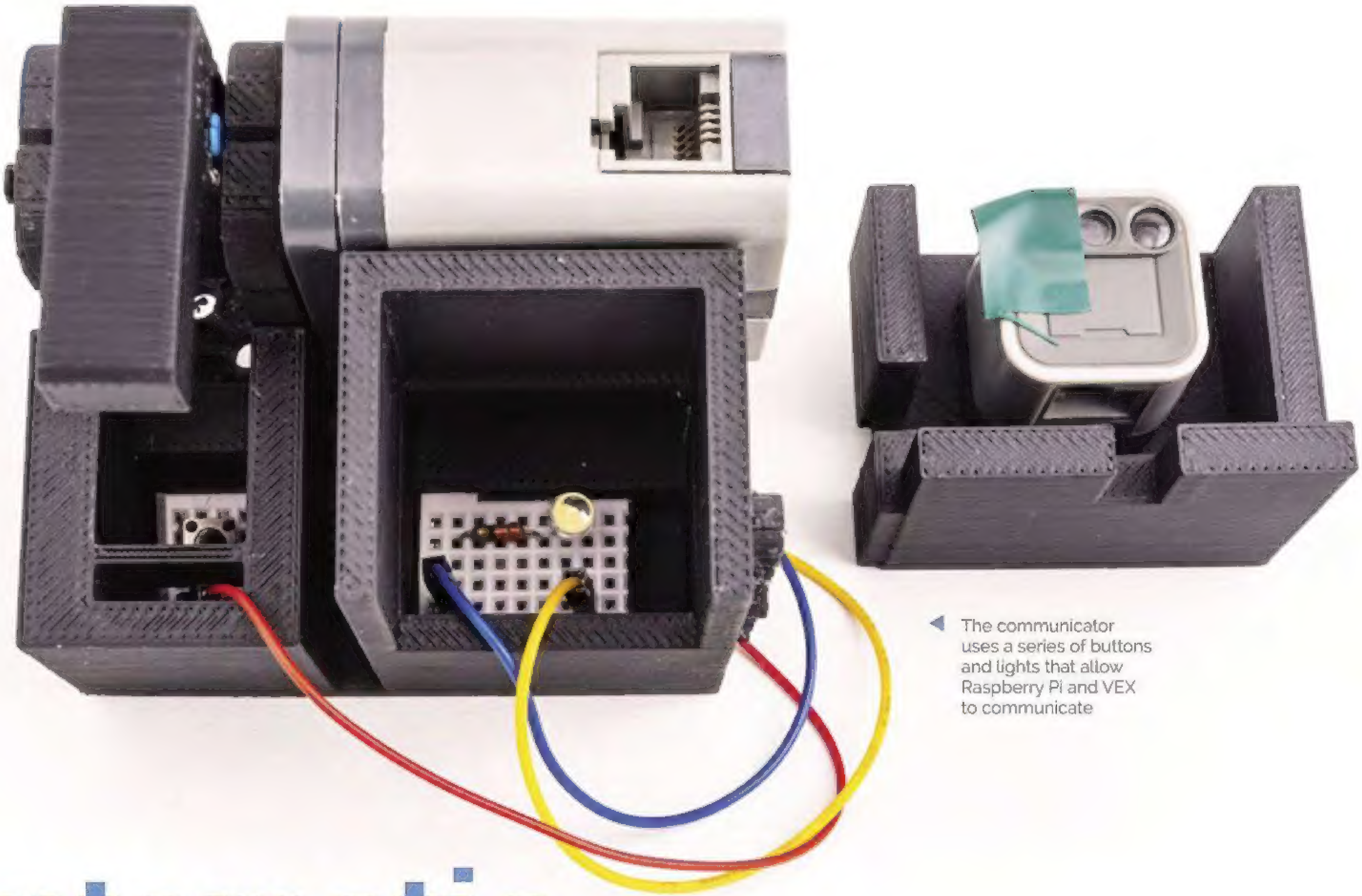
Reviews

Community



Ellora James interview

WIN 1 OF 2 7-COLOUR E-PAPER **DISPLAY HATS** **95**



The communicator uses a series of buttons and lights that allow Raspberry Pi and VEX to communicate

Automatic Seedling Nurturer

When you need to automate plant watering, you could build your own system – or, as **Rob Zwetsloot** finds out, use a programmable robot toy

MAKER

Chloe Alfonso

A high school student who loves tinkering, writing, and playing water polo. She's been using robotics with VEX for years.

magpi.cc/seedling

This month we have a slightly new take on automated plant watering, as created by Engineering Girl, aka Chloe Alfonso. Instead of custom-building a system or hooking up prosumer-grade plant care stuff to a Raspberry Pi, she's built a VEX IQ robot system – a kind of programmable construction block robot – that is powered by Raspberry Pi.

"This specific robot allows Raspberry Pi to communicate with the VEX IQ system to create a plant watering and lighting system," Chloe tells us. "I was trying to raise seedlings and found that sometimes I would forget to water the plants or turn on the grow light. I thought it would be useful to automate the project. VEX – with its motors, sensors, and Lego-like parts – lends itself to making structures. However, there is no internal mechanism for monitoring the time of day. I wondered if I could use a Raspberry Pi with a real-time clock to trigger the mechanism. Raspberry Pi would open up many other possibilities, like accessing the mechanism via the web or recording data points (temperature, humidity, growth) for later comparison."

Ideas on ideas

Just using a Raspberry Pi as a timer, though, is a bit overpowered, but the possibilities of what could be done with it sparked new ideas.

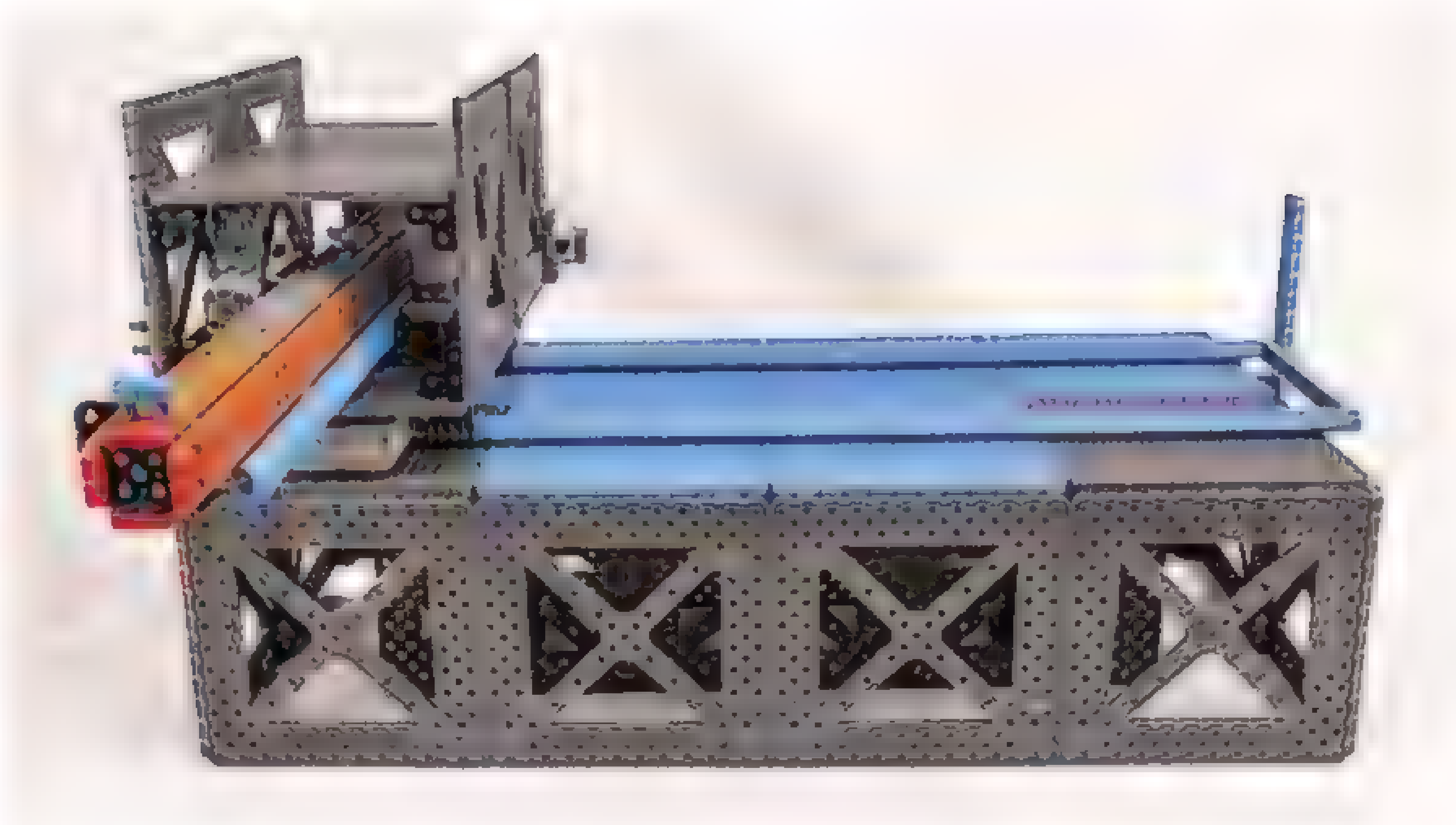
"Initially, the goal was to water seedlings automatically," Chloe reveals. "However, as the idea formed, other benefits of using a Raspberry Pi arose. From an educational standpoint, I learned how to wire a breadboard, an LED light, and a button. Learning to do these tasks with a Raspberry Pi was rewarding in itself, but applying

While the main purpose of Raspberry Pi is to check the time, it can do a lot more with sensors and networking that will be added to the project

The robot and Raspberry Pi are just above the plants, with the robot arm moving an output for a water pump

FACTS

- VEX is not dissimilar to Lego Mindstorms
- VEX has regular robotics competitions for young makers
- Like a lot of robotics projects, it has a huge amount of growth potential
- A 3D-printed part allows for VEX / Raspberry Pi communication
- The water pump is operated by a relay, so be careful if you plan to recreate it



the wiring of the breadboard and peripherals as part of my project was an added bonus."

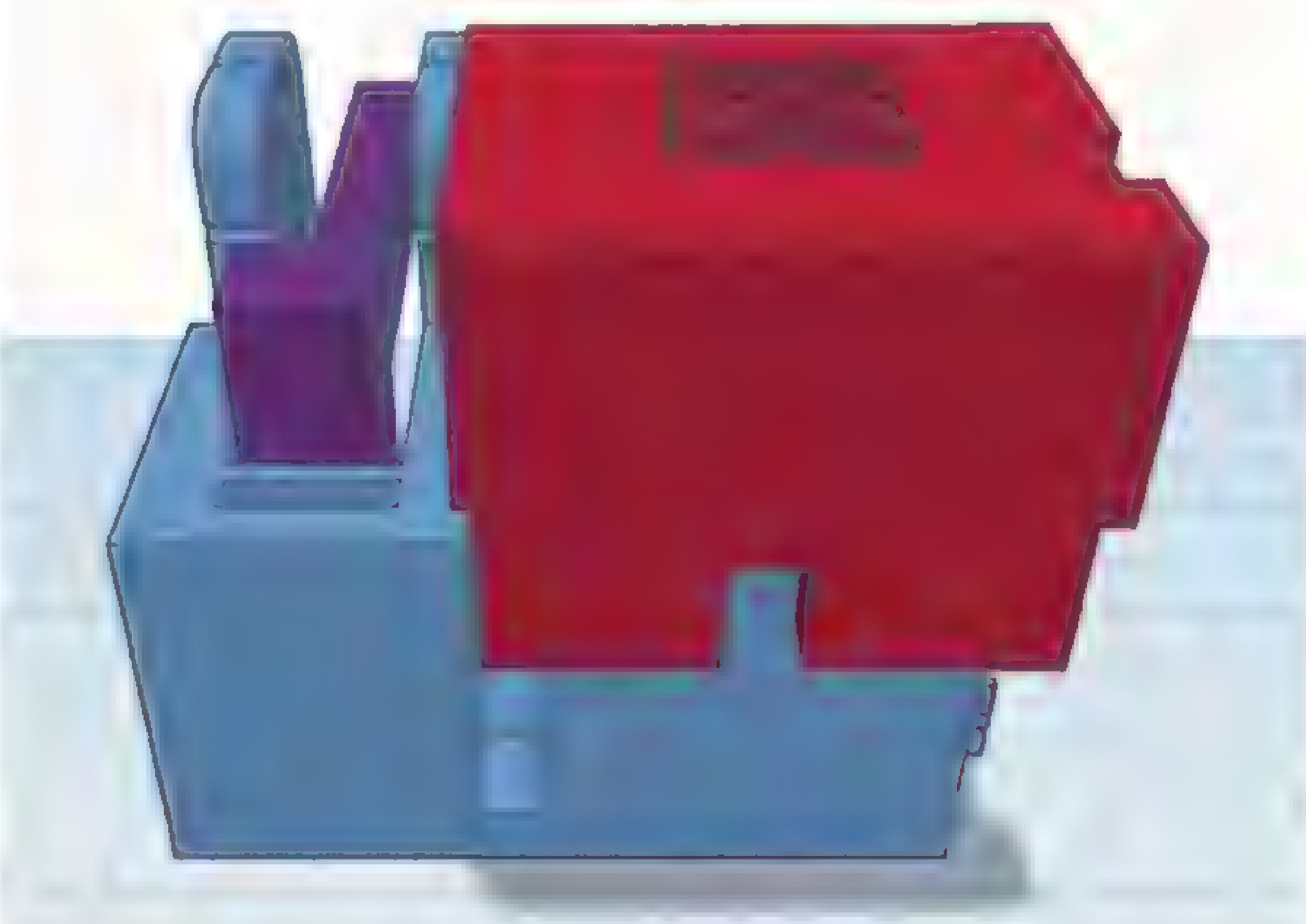
She says there were many reasons for using Raspberry Pi. "For one, with its multitude of sensors, accessibility to networks, and bountiful options for programming languages, [it] provided

> However, as the idea formed, other benefits of using a Raspberry Pi arose

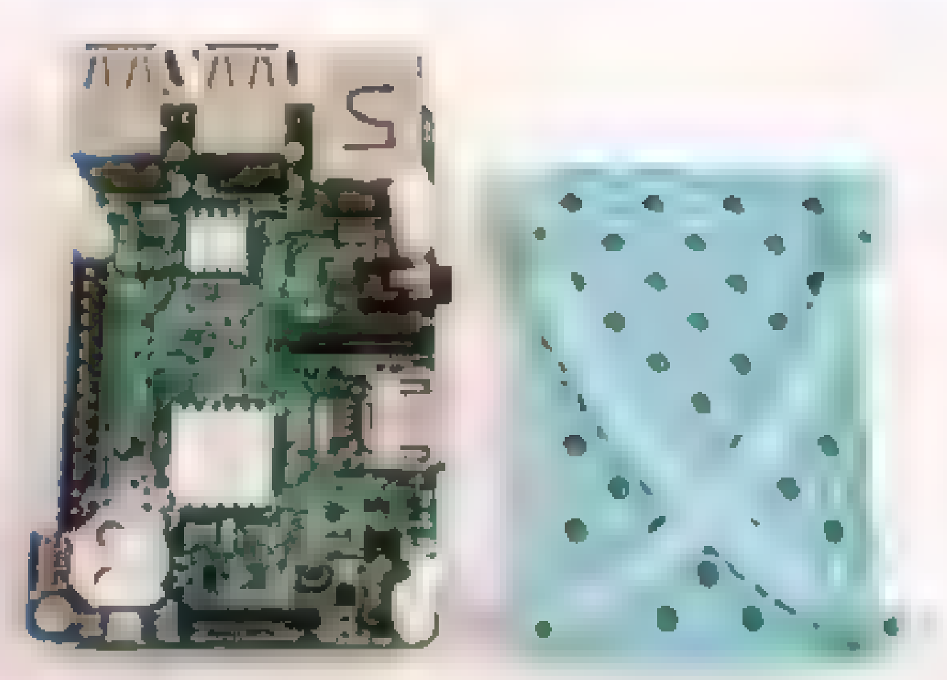
a flexible solution that could not only be used in this project, but whose lessons could be expanded to other projects. This project could have been done more simply with a UART connection using an older model VEX EDR microcontroller. This way, the transmission of information would have then been direct. But by doing the project with

▲ The full robot arm looks wacky but is very functional

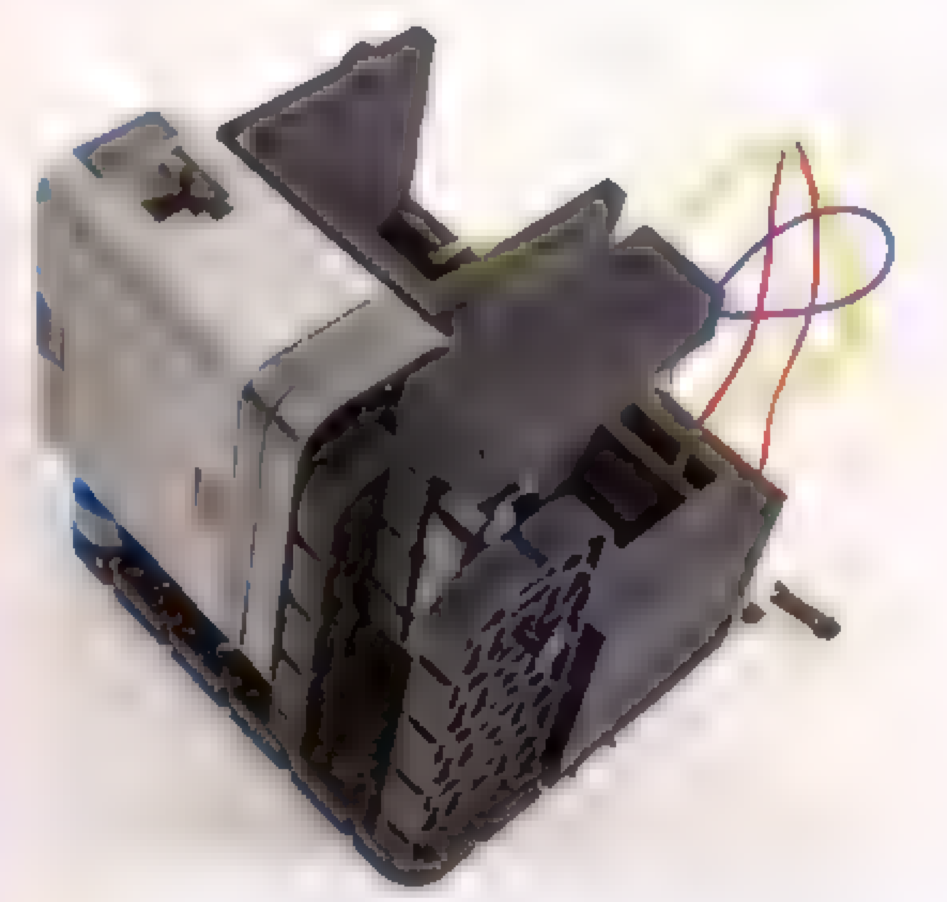
▼ It was all designed in CAD before 3D printing



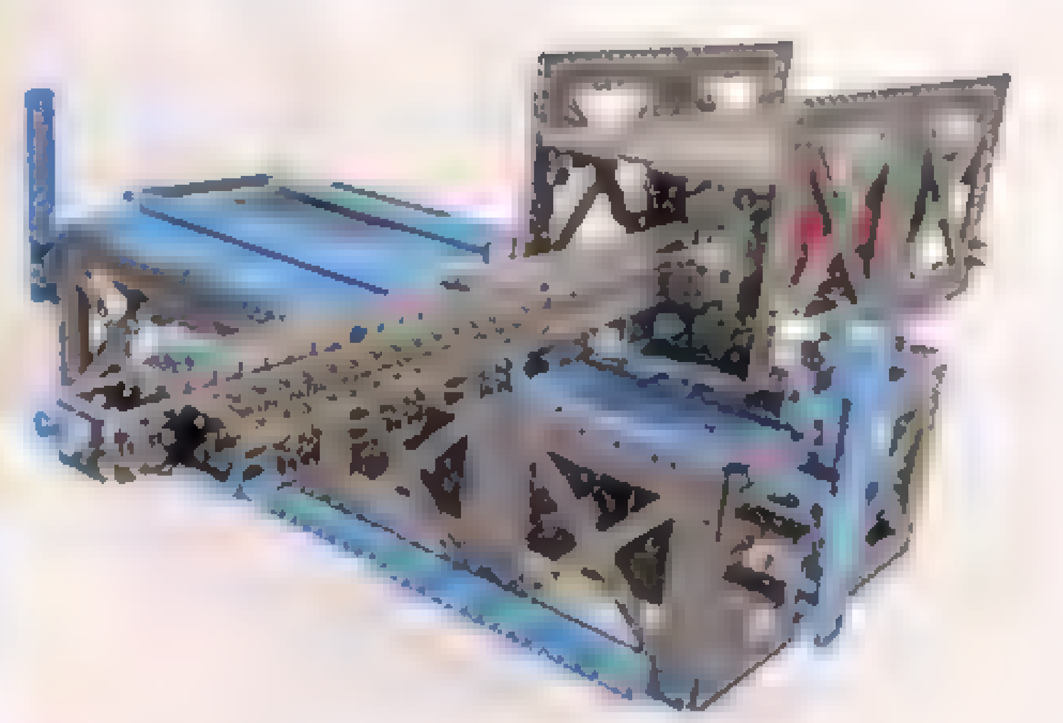
Seedling watering



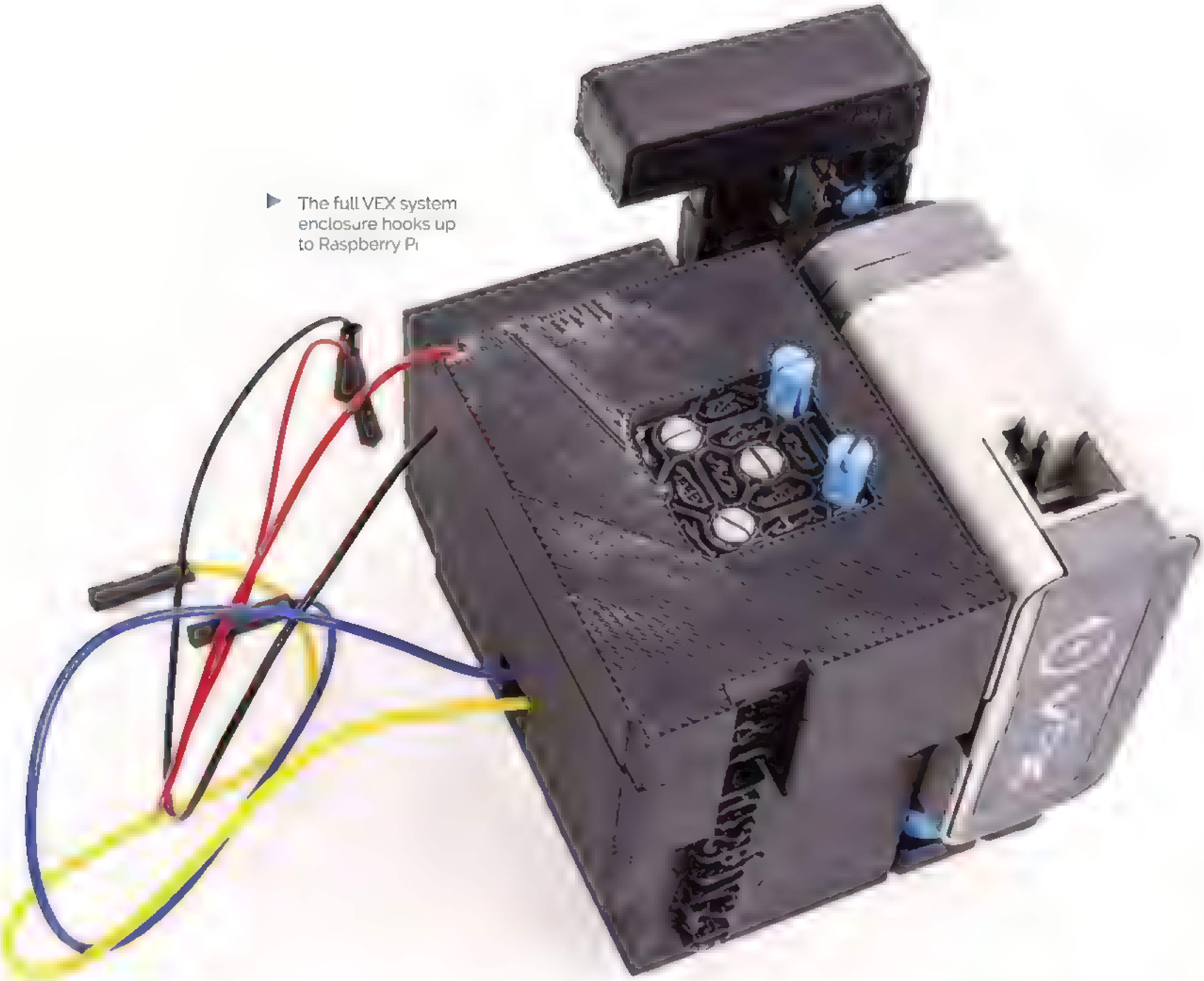
01 Raspberry Pi keeps track of time. When a predetermined time passes, Raspberry Pi triggers an LED light on a breadboard.



02 The LED light is sensed by the VEX IQ microcontroller that will move a watering arm over each seedling



03 As the arm is over a seedling, the VEX IQ microcontroller pushes a button that is sensed by Raspberry Pi and triggers a watering pump to stream water over the plant. The apparatus does this for each plant and then resets.



The full VEX system enclosure hooks up to Raspberry Pi

the Raspberry Pi VEX IQ communicator, I learned about 3D printing, circuits, LEDs, and GPIO pins."

Let them grow

We've seen that Raspberry Pi can help grow plants and such, but can a Raspberry Pi/construction block robot hybrid do the same?

"It works quite well," Chloe says. "There is a YouTube video that shows not only how it works, but a couple examples of seedling growth (**magpi.cc/seedling**). I had to connect the VEX IQ microcontroller to a constant DC power supply. Sometimes, one in 40 waterings, one of the motors would seize up and my apparatus would commit egregious overwatering. This was uncommon, however."

Chloe isn't resting on her laurels either: "I want to learn more about networking. My goal is to create an easy-to-use interface to be able to manipulate the apparatus from distant sites. I also used the Raspberry Pi VEX IQ Communicator to create an automatic pet feeder and water bowl filler. There is a short video on the same YouTube channel of this device. The pet carer is in a more nascent stage of development."

A pump is used to supply water to the system

Commodore 64 Revamp

Be like Stephen Williams and bring an old computer back to life. **David Crookes** takes a look

Stephen Williams

Stephen is an original 1980s retro-computer owner and a jack-of-all-trades who is happy to try new things. When not tinkering, he likes getting outdoors, exploring, and spending time with his family

magpi.cc/c64revamp

As many readers will know, Raspberry Pi can be turned into a brilliant, action-packed retro gaming arcade. Using operating systems like RetroPie, you can easily switch between emulators of many age-old home computers and consoles, and scores of makers have made use of this in various weird and wonderful ways.

In this instance, Stephen Williams has brought a broken Commodore 64 (C64) computer back to life. He's stripped out the original motherboard, replaced it with a Raspberry Pi computer, and used Lego bricks to build the internal housing. It reminds us of Christian Simpson's fantastic Brixty Four project (**magpi.cc/brixtyfour**). But while that sought to create a new C64 case out of Lego, Stephen's project retains the original, iconic 'breadbin' plastic.

Externally, the Commodore 64 Revamp appears identical to the original Keyboard mapping software is used to communicate with Raspberry Pi

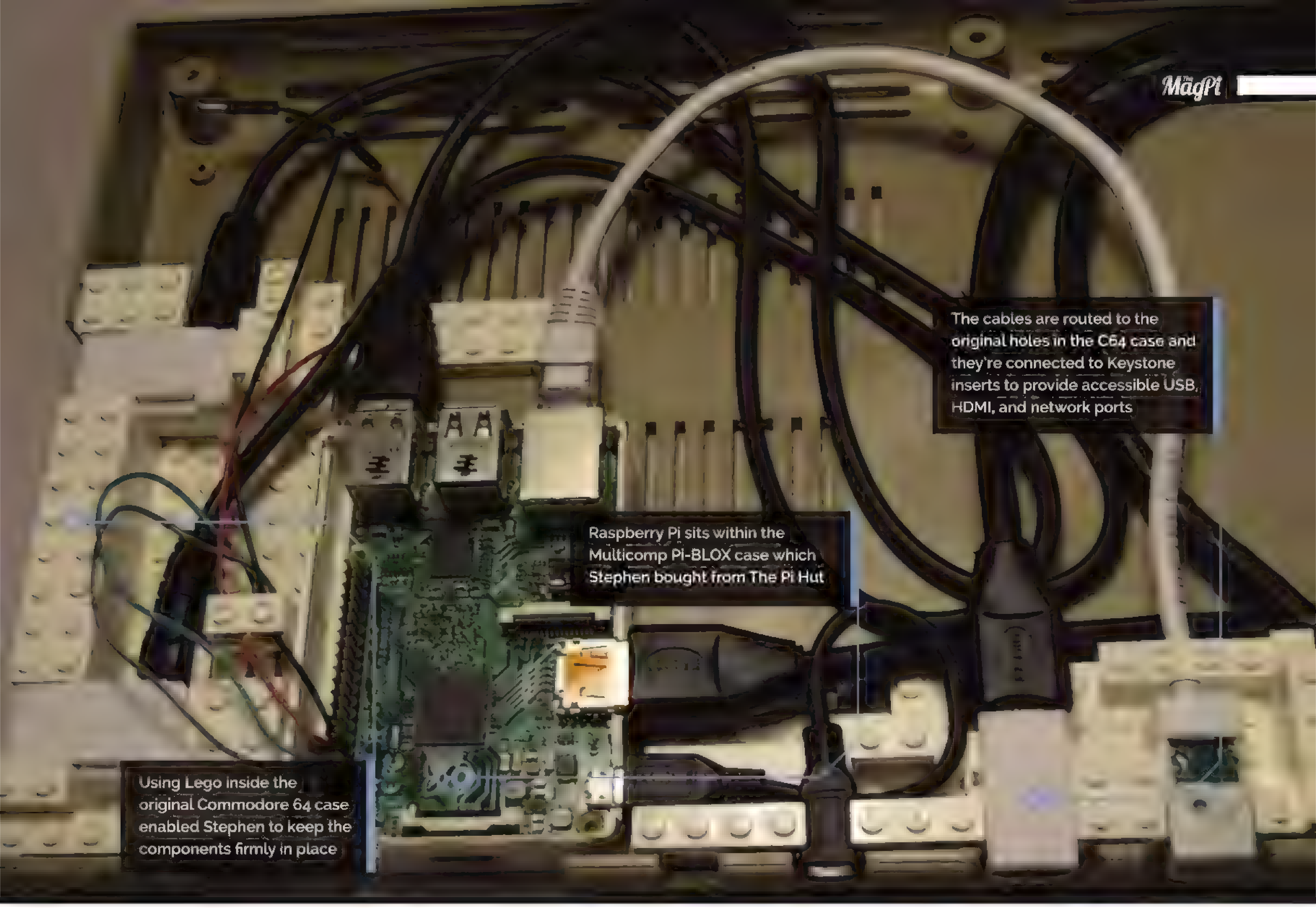
"I've played with Raspberry Pi since the computer first came out, making and discovering new things along the way," he tells us. "I had a truly broken Commodore 64 that I felt would benefit from a new lease of life. My project blends retro with a modern twist, and brings together some of the things I've liked to play with over the years – notably Lego, the C64, Raspberry Pi, and Arduino."

Building the project

Stephen turned to Lego for a practical reason. "I didn't have access to a 3D printer, and Lego has so many different pieces that it's really versatile to experiment with," he says. "I already had a Lego Raspberry Pi case, so it seemed to make sense to build around it. I wasn't sure what the fit would be like inside the C64, so I bought a mix of pieces from a market stall to see what would work."

He says it was important to build a solid base inside the C64 and position the USB, HDMI, and other ports in the right places. "I didn't want to modify the computer's case in any way and Lego helped me to do all of this." The rest of the build was rather straightforward and involved inserting a Raspberry Pi computer into the case so that the ports were accessible, inserting a microSD card with RetroPie installed on it, and connecting to the C64 keyboard.

To do this, Stephen used an Arduino Micro. "It provides the mechanism to get a fully working C64 keyboard for Raspberry Pi," he explains. "The basic idea is to scan the pin readings on the Arduino which are connected to the row and column pins on the C64's matrix keyboard. Using the Arduino



The cables are routed to the original holes in the C64 case and they're connected to Keystone inserts to provide accessible USB, HDMI, and network ports

Raspberry Pi sits within the Multicomp Pi-BLOX case which Stephen bought from The Pi Hut

Using Lego inside the original Commodore 64 case enabled Stephen to keep the components firmly in place

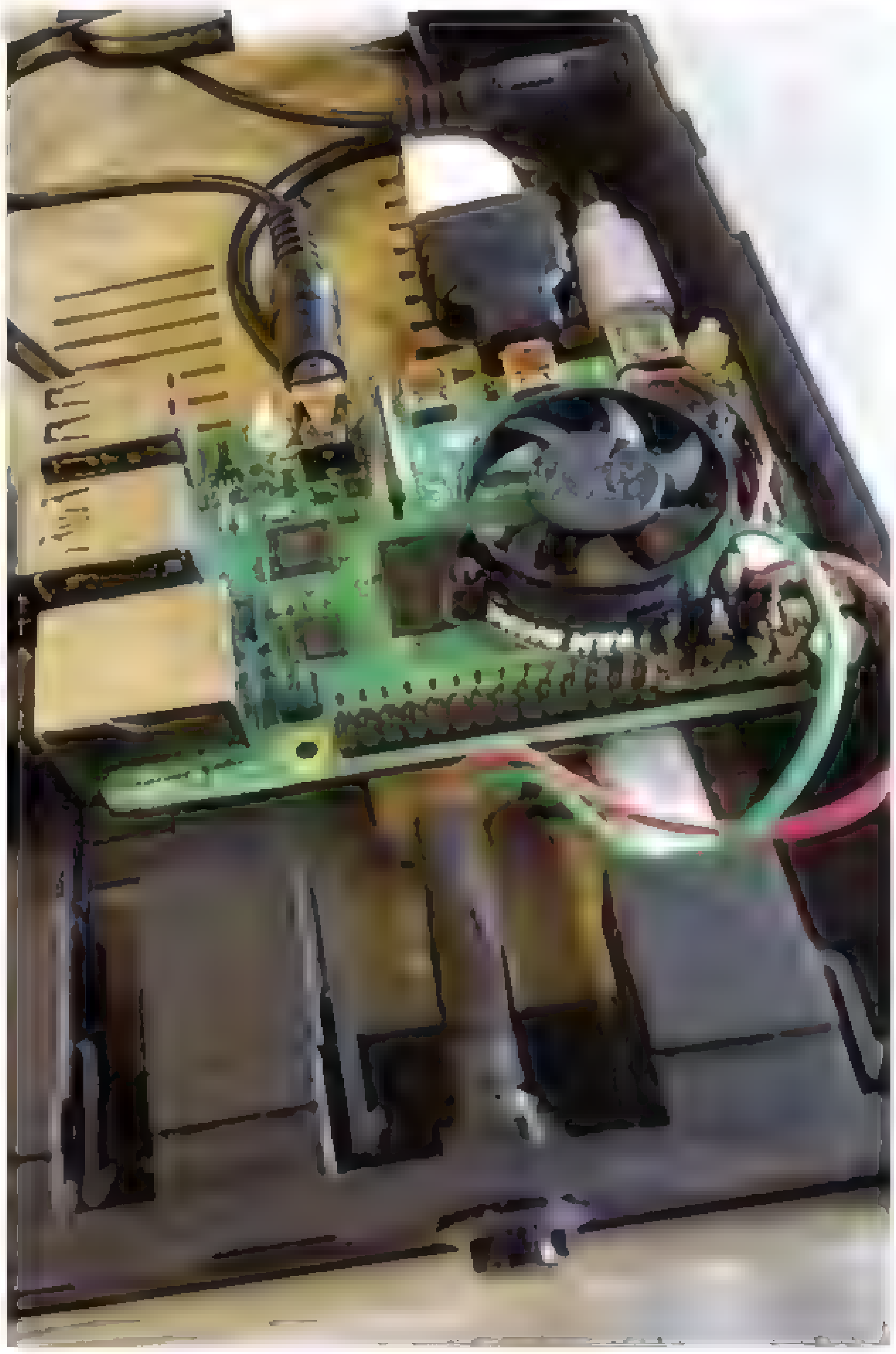
> Lego has so many different pieces, so it's really versatile to experiment with

software libraries, the row and column pins are scanned, and the mapped keystrokes are sent to the computer connected to the Arduino via USB."

Pulling it apart

Since creating this project, Stephen has acquired a 3D printer. As such, he's been replacing the Lego using printed parts, again for practical reasons. "When Raspberry Pi 4 came out, I wanted to use it but because I needed to install a fan, I couldn't use the Lego Pi case any more," he says. "The 3D build means I've been able to get closer to the original Commodore 64 regarding the location of the power socket and switch."

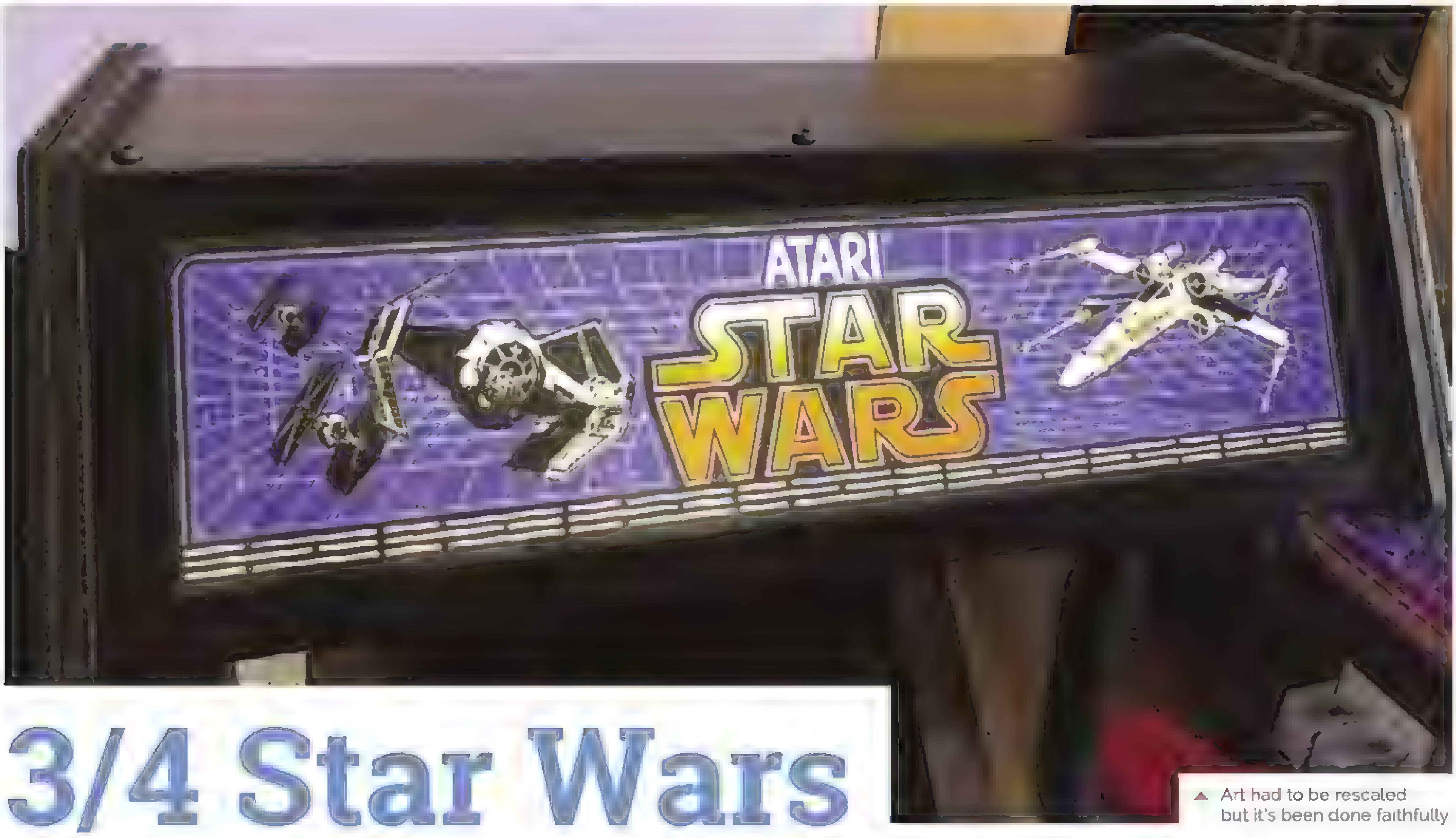
Even so, he's not always entirely faithful to the C64. "Since RetroPie brings many emulators into one place, it's been a bit surreal playing a Spectrum game with the C64 sitting in front of me, but I've become used to it. RetroPie is also easy to extend to include things such as homebrew programs, media players, and bespoke themes. It's been fun to dabble with these too."



The 3D printed build for comparison. Different cable requirements are also needed for Raspberry Pi 4 setups due to it using micro HDMI and USB-C ports

QUICK FACTS

- No coding or electronic skills are needed
- The original C64 case is not modified either
- It's a great way to revive a broken C64
- You can even use USB joysticks
- Try applying the build to VIC-20 and C16 computers



▲ Art had to be rescaled but it's been done faithfully

3/4 Star Wars Arcade Cabinet

Why pay over the odds when you can build an accurate replica, and have fun doing it? **Rob Zwetsloot** switches off his targeting computer to have a look

James Milroy

A postman who loves making and fixing things in his free time, and spent his childhood in arcades

@james_milroy

Getting the arcade machine of your dreams gets a little harder every day, especially the older they are. Making one, however, is always possible if you have the right skills and a Raspberry Pi.

"My project was to build a replica, or as close as I could reasonably manage, of the Atari Star Wars arcade cabinet," James Milroy tells us. "I really wanted to build a cockpit as that's what I played on in the eighties, but sadly I didn't have the room to house it, so the compromise was to build a stand-up cabinet instead."

Even then, the standard cabinet has a lot of detail, and James really nailed the look of it. Why build it from scratch, though? "Initially, I had toyed with sourcing an original cabinet and restoring it, but soon gave up on that idea after finding it nigh on impossible to source a cabinet here in the UK," James explains. "Almost all cabinets for sale were located in the USA, so they were out of the question due to the high cost of shipping. Atari only made just over 12,500 cabinets worldwide, so their rarity meant that they commanded top dollar, effectively putting them out of my price range. It was at this point that I decided that if it was going to happen, then I would have to make it myself."

Making a cabinet is hard enough, but the control system would have to be an original Atari yoke. "The Atari yoke is considered the 'holy grail' of controllers and, again, is very hard to find," James says. "My prayers were answered in October 2018 when a thread on a forum I was subscribed to popped up with a small Utah-based startup aiming to supply replica yokes at a realistic price to the arcade community. I grabbed two of these (one for my friend) and the project was on."

Good feeling

When it came to actually emulating the game, for James there was only one choice: "My decision to go with a Raspberry Pi was a no-brainer really. I had previously made a bartop cabinet using a Raspberry Pi 3 and RetroPie/EmulationStation which I was really pleased with. So I had a

The cab is made with MDF, and includes the original art

The yoke is an accurate replica that connects to Raspberry Pi via USB

The riser was a custom build by James that emulates lights from the films

- The original game came out in 1983
- The same year as *Return of the Jedi*
- It used special vector graphics to emulate 3D space
- A Picade X-HAT handles everything but the replica yoke
- A lot of custom work was done to downsize stuff to 3/4 scale

▲ The coin slot is fake, and just requires button presses to add credit

▲ 3D-printing the mouldings and details is easier than recreating them with vacuum forming

> Overall, I'm really pleased with the way the cabinet has worked out

▲ Audio and lights are easier to install with modern tech

platform that I already had experience with and knew was more than capable of emulating the one game I needed to run. Besides, the simplicity and low cost of the ecosystem for Raspberry Pi far outweighs the extra expense and effort required going down the PC route."

With a custom build and emulation, authenticity of the gameplay experience could be a bit off. However, that's not the case here. "I think that it plays just like the real arcade machine mainly due to the inclusion of the replica yoke controller, and adding your credit by pressing the button on the coin door," says James. "Ideally a vector monitor or a CRT would go a long way to making it look just like the original, but a reasonable representation is possible on an LCD using shaders and anti-aliasing. Gameplay does seem to get really hard really quick, though; this could be due to an imperfect emulation, but is more likely due to my reactions having dulled somewhat in the last 38 years!"

Always in motion

While the current build is amazing as it is, James does have some ideas to improve it. "Overall, I'm

really pleased with the way the cabinet has worked out," he says. "I will be replacing Raspberry Pi 3B+ with a Raspberry Pi 4 to enable me to run a newer version of MAME which will hopefully offer a better emulation, sort some audio glitching I get with my current setup, and hopefully enable some graphical effects (such as bloom and glow) to make it look more like its running on a CRT."

▲ The wooden parts are laser-cut for added precision

Accurately replicating

01 "Luckily… I managed to have a play on an original cabinet at the Game On exhibition at Glasgow Braehead Arena. Armed with my trusty tape measure, I got some vital measurements from the real cabinet and after that, everything fell into place."

02 The cabinet was designed and laser-cut from MDF. There's some steel in the construction, as well as tempered glass over the screen. Mouldings are 3D-printed, with the help of a graphic artist who also helped rework graphics to fit the smaller size. Lastly, a riser is constructed to make sure it's tall enough to play.

03 "The workings were simple when it came down to it: Raspberry Pi 3B+ with Pimoroni Picade X HAT. This gives us a power switch, audio amp, buttons, and a joystick if necessary. The replica yoke is interfaced with a USB adapter from the same company. It allows us to use the yoke with the original Atari connector. Software is RetroPie and I'm currently emulating on AdvMAME."

▲ Dede can bring the smart chalkboard to life with a tap on her phone; coming up with witty missives takes a bit longer

LCD Chalkboard

A cartoon burger joint inspired this fun Raspberry Pi Zero creation. **Rosie Hattersley** hears how

Dede Mitchell

New Orleans-based Dede Mitchell provides logistical and administrative wizardry at non-profit **or-nola.org**. In her free time she builds one-of-a-kind "tech-infused" custom costumes, wigs, and hats

magpi.cc/milliner

Dede Mitchell's colleagues are so used to seeing her in wigs, costume, and outlandish make-up, they barely bat an eyelid. In a city full where "we costume all the time", dressing up is practically de rigueur, but native New Orleaner Dede loves to add a tech twist. Her latest, the LCD Chalkboard (**magpi.cc/lcdchalkboard**), is her first ever Raspberry Pi project.

Making costumes for herself and others under the guise of the Maladjusted Milliner takes up much of her evenings and weekends, while Dede's day job is as an intra-office administrator and logistics expert at Operation Restoration (**or-nola.org**), a non-profit organisation where "an awesome group of women provide assistance to women and girls affected by incarceration".

Food for thought

Faced with more downtime than she knew what to do with during lockdown, Dede was keen to keep her mind active and decided to learn about Raspberry Pi. After "binge-watching" the cartoon TV show *Bob's Burgers*, an idea for her first Raspberry Pi project began to emerge: the LCD Chalkboard was inspired by the burger of the day displays on the show, as well as "the many smart mirror projects online".

Dede is a confident and experienced wearable tech maker who builds "anything that comes to my mind. Usually I build wearables [for New Orleans' parades including Mardi Gras], so I like smaller microcontrollers such as Trinket and Feather." Her love of LEDs and blinking lights started when Dede was helping organise a STEM event for kids and was introduced to microcontrollers, Adafruit, and Arduino. However, coding didn't click until she picked up a Raspberry Pi Zero W and started learning bits of Python and MicroPython. "For me it is intuitive and just makes sense," she reports.

"I love Raspberry Pi Zero W because, unlike a regular microcontroller, Raspberry Pi Zero W is a minicomputer. It allows me to build some of the projects that only live in my head right now. Interactive costume pieces are buzzing around in my head."

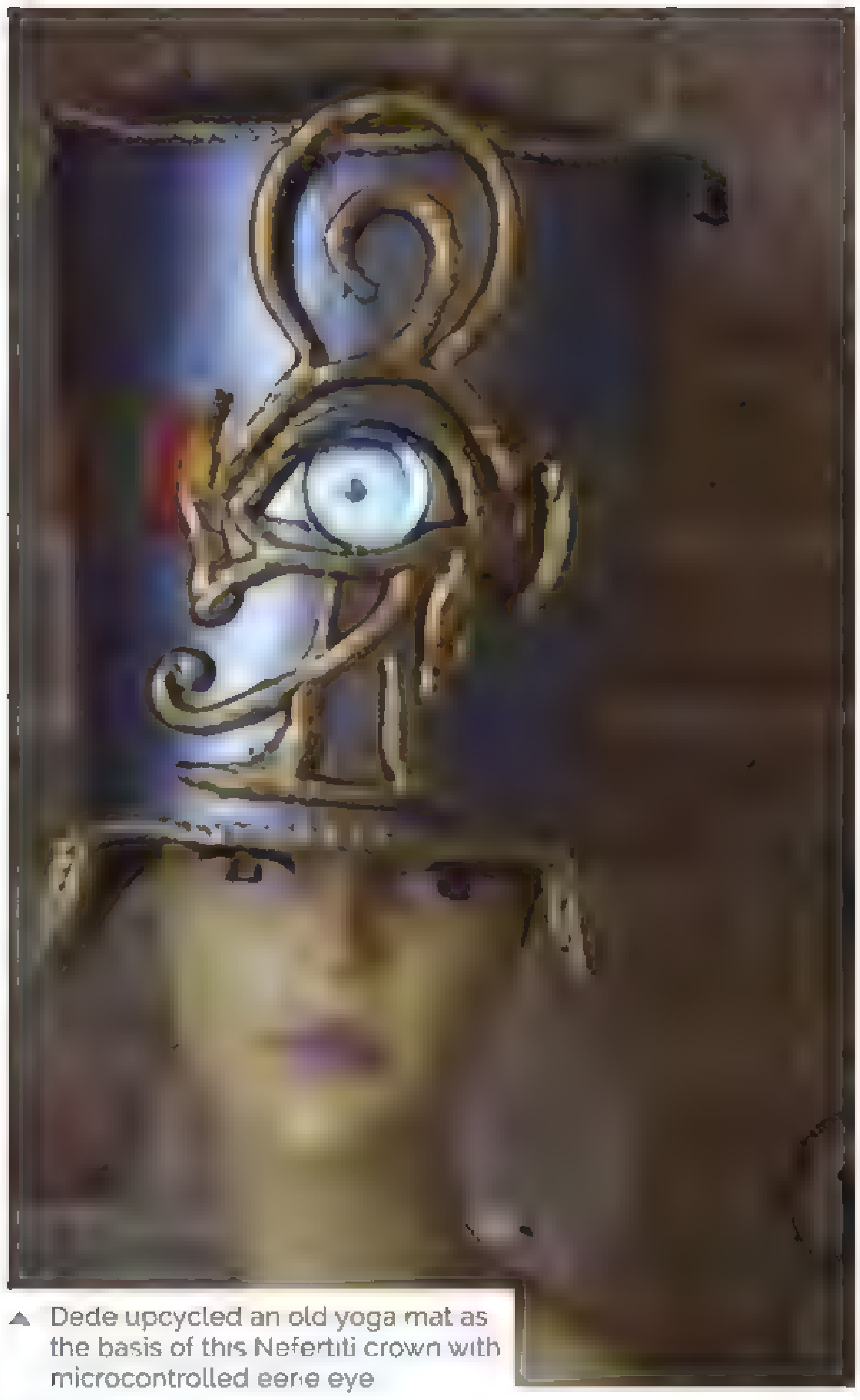
As well as having magic mirror-like features, the LCD chalkboard references maker Dede's love of cartoon TV show *Bob's Burgers*

She has no intention of giving up writing bad puns

Dede made the chalkboard frame, so this is her first woodworking project, as well as her first Raspberry Pi project

QUICK FACTS

- The *Bob's Burgers* TV show inspired this build
- Dede says her board is likely "to contain many bad *Star Trek* puns"
- The board has a secret compliments mode!
- These provide pick-me-ups after an exhausting day…
- …and may also reveal the whereabouts of the chocolate stash!



Dede upcycled an old yoga mat as the basis of this Nefertiti crown with microcontrolled eerie eye

Dede's 'tech-infused' Infinity Space Helmet features a trio of infinity mirrors and a microcontroller

Warning!
Drill and saw

This project uses an electric drill and saw Be careful when using sharp tools.

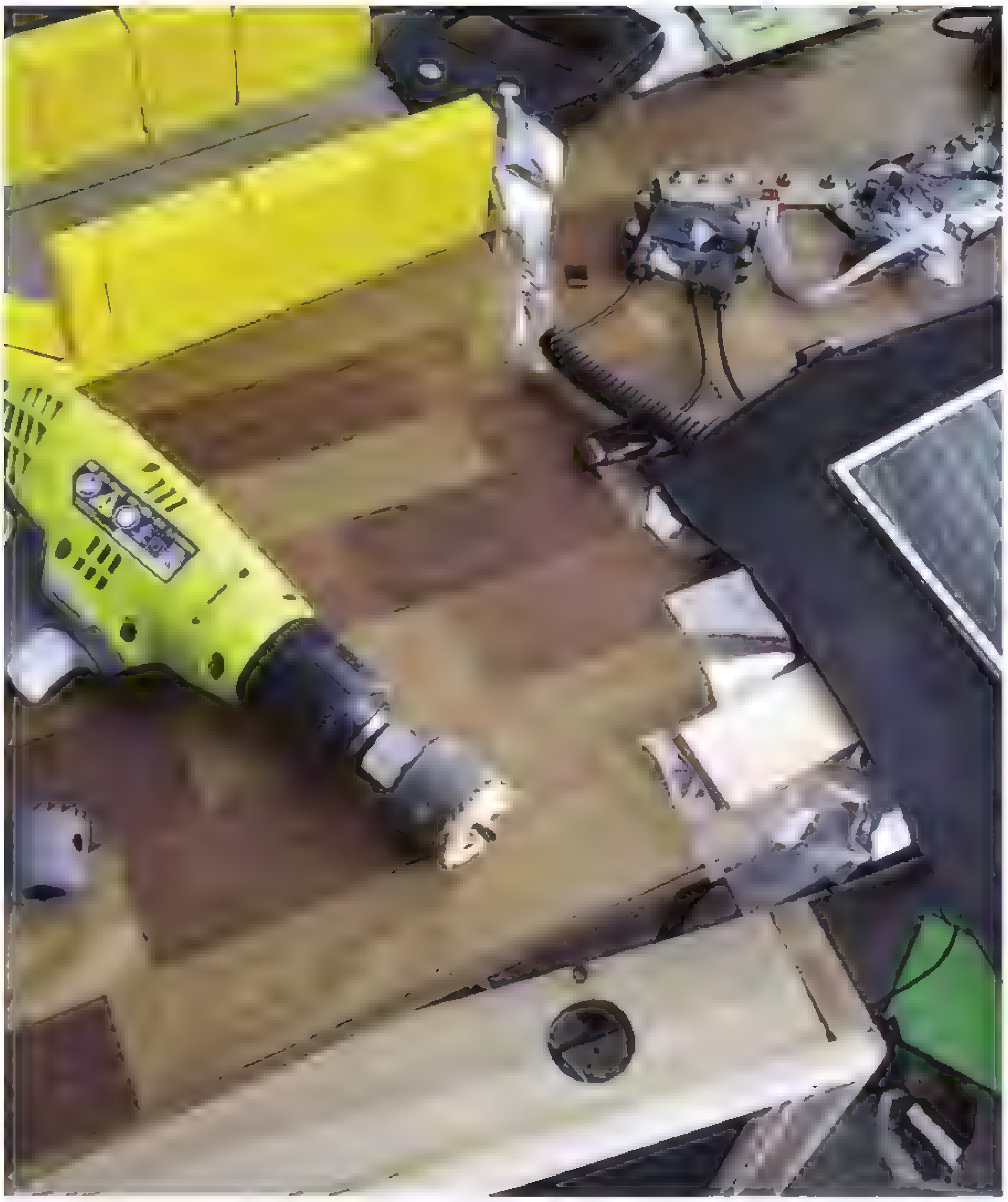
magpi.cc/drillsafety

> I love Raspberry Pi Zero W because, unlike a regular microcontroller, Raspberry Pi Zero W is a minicomputer

Adaptable attitude

By the time she began designing her chalkboard, there were three Raspberry Pi Zero W boards on hand. She needed to adapt the smart mirror example builds she found online as most were not for Raspberry Pi Zero. "But I did find a couple of guys that got it to work, so I cobbled together information from their sites and I was off to the races!" she says.

However, using Raspberry Pi Zero meant she had to use it headless. At first she was intimidated by the command prompt, but having followed a useful guide she found online, it's "so ordinary for me now, I



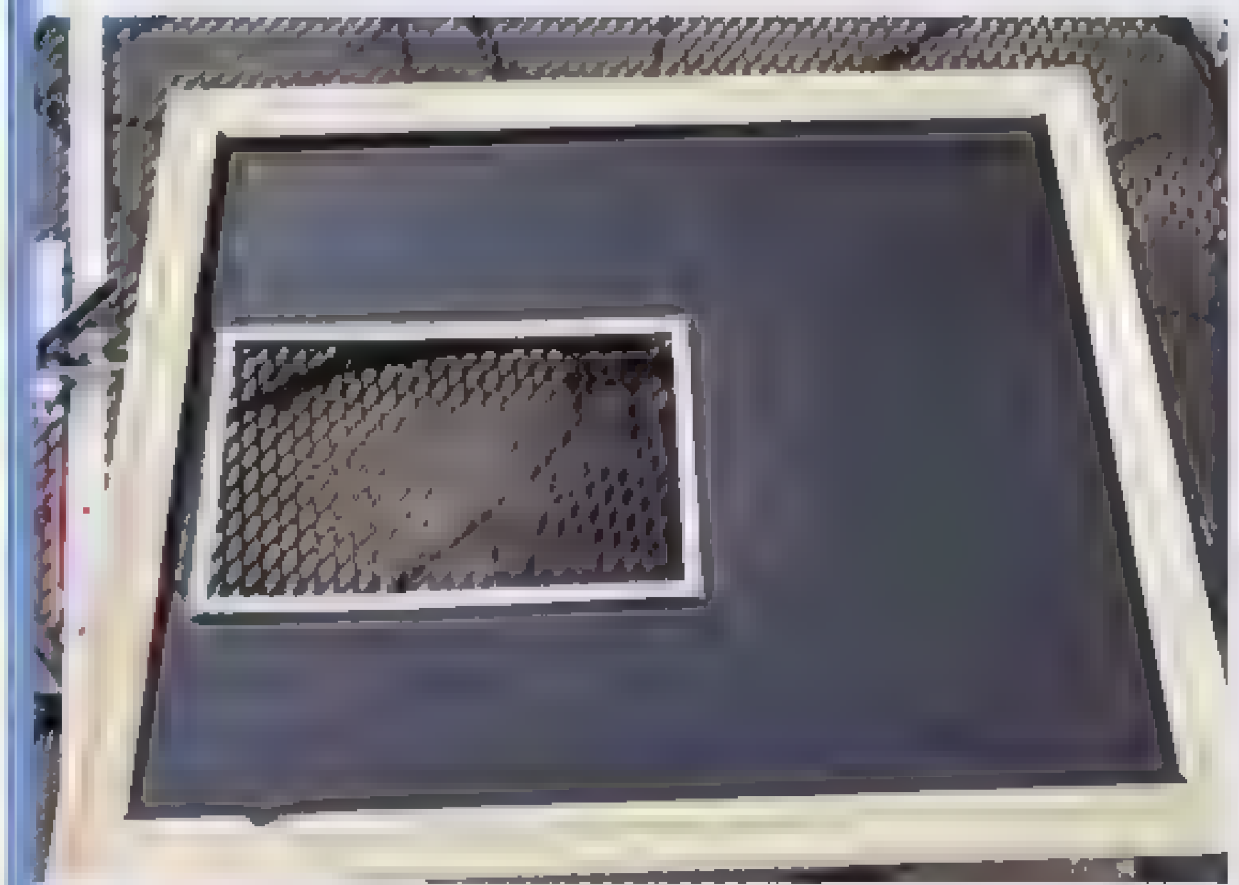
Dede used YouTube to get tips on neatly sawing a hole in the plywood to fit the LCD, which she upcycled from an unwanted laptop

occasionally reboot or shut down my sign with my phone, from my bed."

The chalkboard design is all Dede's. She decided to use whatever tools were available, and learnt to use a jig-saw and scroll-saw for this project. Once she was happy with her sketches and the design, the build process took around two weeks' worth of evenings and weekends. To keep things simple, she stuck with the existing Raspberry Pi OS wallpaper, but intends to customise this with her holiday photos once her coding skills allow.

She's largely happy with how the project turned out and with Raspberry Pi Zero W, but will eventually upgrade it to Raspberry Pi B+. "More processing power allows for better interaction," she reasons, and Dede is keen to add face- and voice-recognition by adding speakers and a camera. Along with learning "some basic Java script, and basic Linux commands," she's going to invest in a framing square. "I couldn't find mine during the build and decided to wing it!"

Build an LCD chalkboard



01 Setting up an LCD chalkboard is broadly similar to creating a magic mirror, for which detailed instructions are on GitHub (**magpi.cc/magicmirrorgit**). Apply chalkboard paint to one side of your plywood chalkboard, then cut out a hole the correct size and dimensions for your LCD. Measure and fit the frame.

02 Lay out and connect Raspberry Pi, AV controller board, SD card, and display power cables. A four-gang power strip helps organise power supplies

03 Use `sudo apt-get` to install the magic mirror image to your SD card, insert into Raspberry Pi, and power everything up.

Terminator HK Tank

Scary futuristic robot alert! **Phil King** delves into the makings of this intelligent automaton

Michael Darby

Growing up watching sci-fi shows like *Star Trek*, Michael became obsessed with technology – robots, AI, holograms, etc So, he's always wanted to try to make his own.

314reactor.com

Featured in the first two *Terminator* movies, Skynet's HK Tank is a fearsome military machine from a post-apocalyptic future – 'HK' is short for 'Hunter-Killer'. Standing several storeys high, this heavily armoured robot would trundle forward on caterpillar tracks, its bright searchlight probing the area for humans to blast with plasma cannons.

Fortunately for humankind today, Michael Darby's DIY robot version of the HK Tank (**magpi.cc/hktank**) is a lot less deadly, firing beams of light from RGB LEDs. "I have always been a huge fan of the *Terminator* films and I've always wanted to recreate the robots from them," he tells us. "The tracked chassis [a DFRobot Black Gladiator] was perfect for this – it's not dangerous, but it may bump into someone. The lights on it are also kind of bright."

Enemy detection

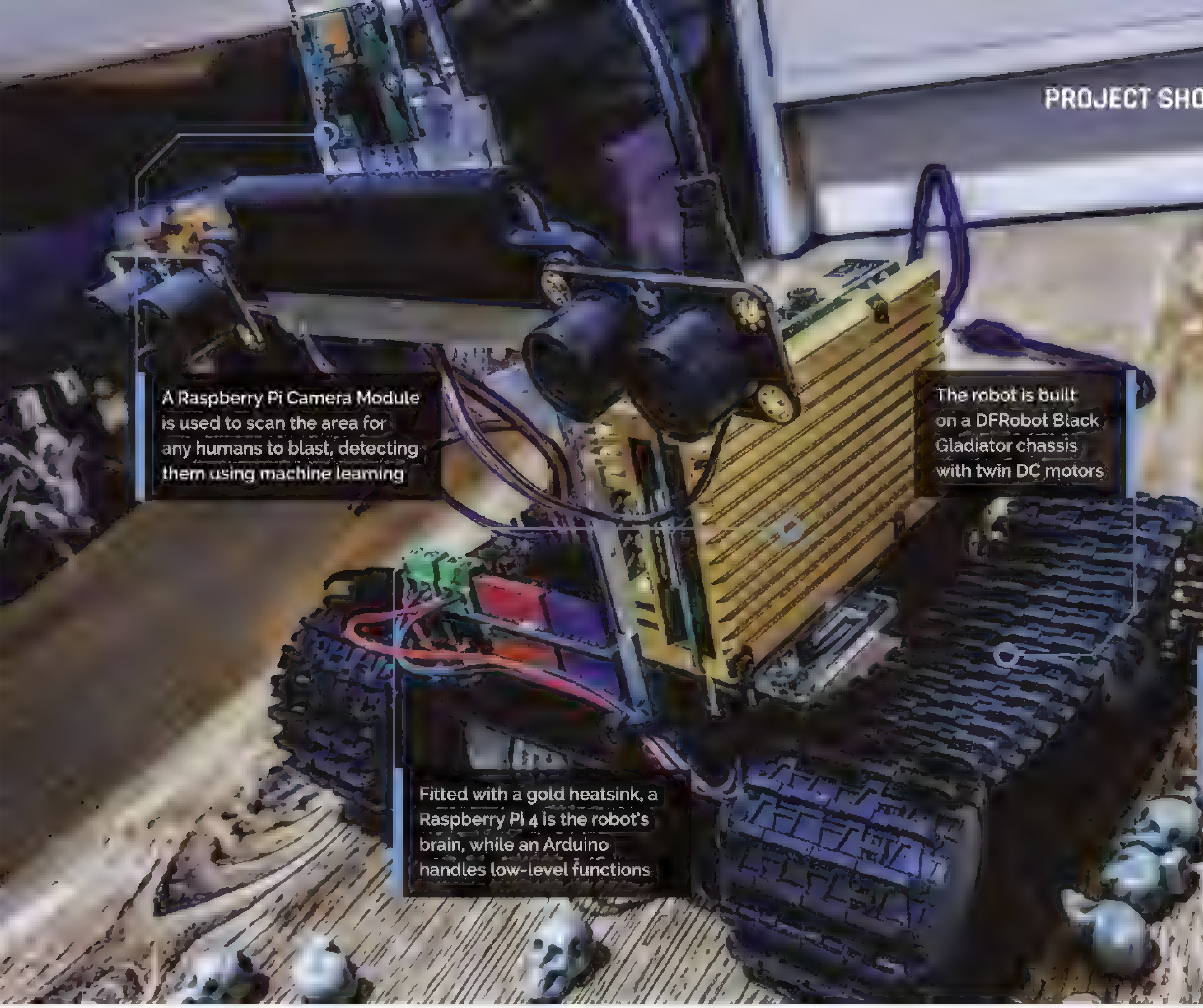
Not only does Michael's robot look the part, it has numerous sensors to help it navigate, along with a Raspberry Pi Camera Module for spotting humans and objects. If a human is detected, 'purple plasma' – or rather, a flash of the RGB LEDs – is fired at them.

Mounted to one side, with a shiny gold heatsink, a Raspberry Pi 4 is the brain of the robot, handing the TensorFlow machine learning and AI functions. "It's using a pre-trained model, so I think it has a good few thousand things at least it can detect," says Michael. "In future I will look into getting it trained on more things."

> Maybe it's just as well it's only firing light beams

The robot's Raspberry Pi also communicates with an Arduino which handles the low-level functions, such as controlling the two DC motors and reading no fewer than five ultrasonic sensors used for obstacle avoidance. "There are probably other sensors I could have used," notes Michael, "but I thought these would be the easiest to

While the Arduino and motors are powered by six AA batteries, a high-capacity power bank is used for Raspberry Pi



A Raspberry Pi Camera Module is used to scan the area for any humans to blast, detecting them using machine learning

The robot is built on a DFRobot Black Gladiator chassis with twin DC motors

Fitted with a gold heatsink, a Raspberry Pi 4 is the robot's brain, while an Arduino handles low-level functions

- Raspberry Pi and Arduino communicate via serial
- Find all the code on GitHub: **magpi.cc/hktankcode**
- Michael's previous projects include *Terminator* glasses and a robot head
- You may also recall his Windows Watch from issue 57 **magpi.cc/57**
- He plans to improve his Raspberry Pi Chatbot project using Docker

implement. There are two on the front to make sure it doesn't hit a wall (and also help with the aesthetic) and the others are all in place to prevent it from hitting things behind it, or driving off of the edge of things."

Join the queue

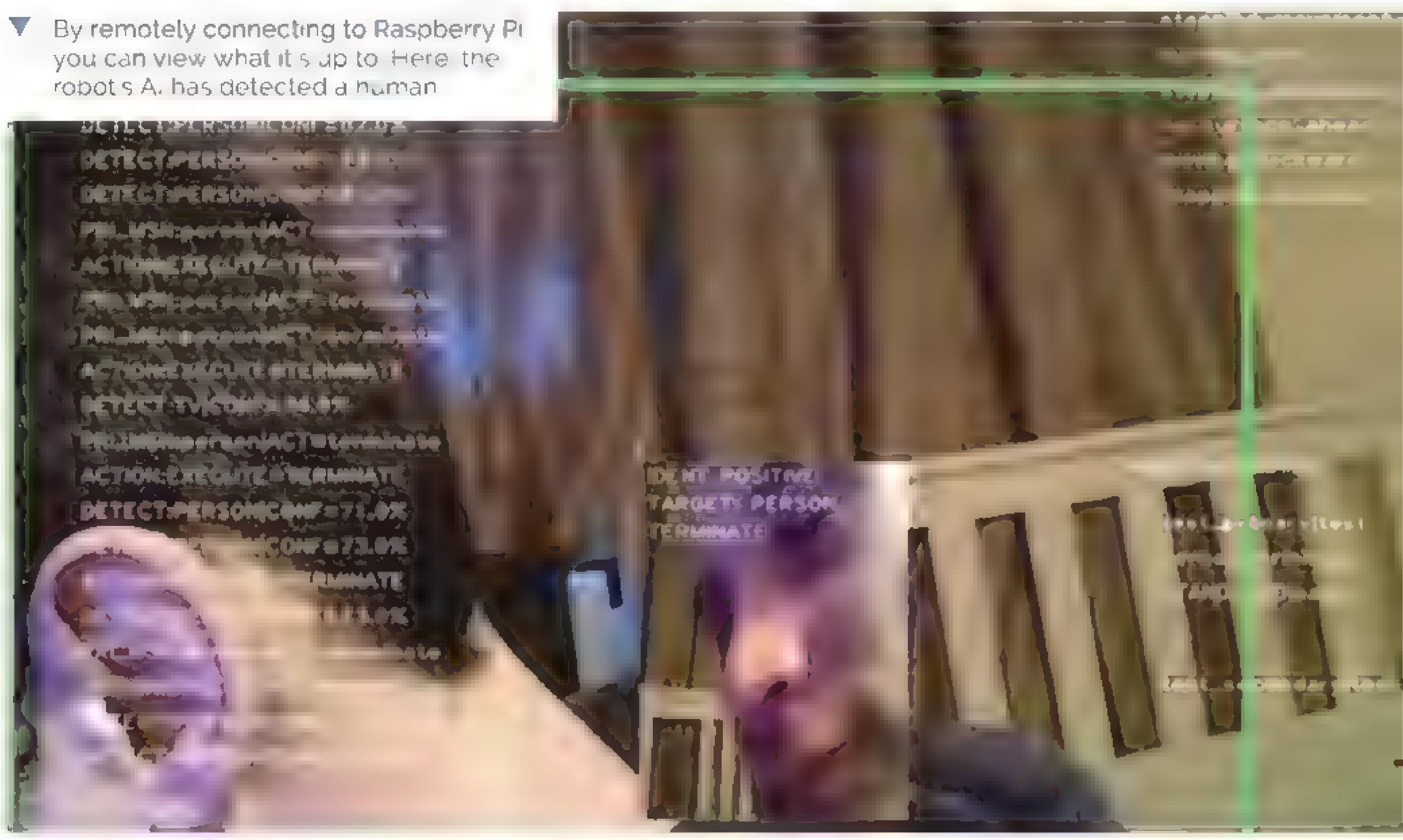
With the project taking "a good few months" to develop, Michael says the most difficult was sorting out the code – written in Python – and getting it to do what he wanted. "This project had some really new stuff I was trying, such as the event queue to allow asynchronous processing; this is where it really got complex."

Rather than handling one function at a time, the code is based on an event queue. All functions are threaded and running simultaneously, dropping their results or requests into the central queue. This prevents conflicts and is based on priority for the functions: "If there is a module that requires something to happen sooner, like some kind of priority action for moving to a location, or recognising a person, it will be handled before a lower-level action such as processing an object that isn't a mission parameter."

Mission parameters are stored in YAML files rather than hard-coded. "You can essentially give it commands via a YAML file that contains all of its standing orders and configurations," explains Michael. "From here it can be set to identify certain things and react in a certain way, such as spot a human and light up the 'plasma turret' RGB LEDs."

While the robot has received positive feedback from the community, he's planning to make improvements, such as training the model on specific faces and "allowing for more complex patrol patterns and 'missions' and make it more of an autonomous bot." Maybe it's just as well it's only firing light beams.

▼ By remotely connecting to Raspberry Pi you can view what it's up to. Here, the robot's A.I. has detected a human



Raspberry Pi Refrigerator

Your Raspberry Pi computer can look after the delicious pies in the freezer, as **David Crookes** discovers

Rick Kuhlman

MAKER

Rick is Head of Product for the IoT dashboard platform Initial State by Tektronix (**initialstate.com**) He has more than 15 years experience in IoT and SaaS leadership roles working on smart locks, mobile apps, connected car and video

magpi.cc/rpfridge

If the power in your home is cut, there's a real risk any food or other items placed inside a fridge or freezer will become spoiled. According to experts, refrigerated perishable food needs to be thrown out after four hours without electricity. Anything kept in the freezer will last 48 hours – but only if it's full.

It's a good idea, then, to create an alert system that will inform you if your fridge or freezer is suffering a drastic drop in temperature. One such solution has been created by Rick Kuhlman, who has sought to save the world's frozen pies (and other foodstuffs, of course) from being destroyed by making good use of a Raspberry Pi computer.

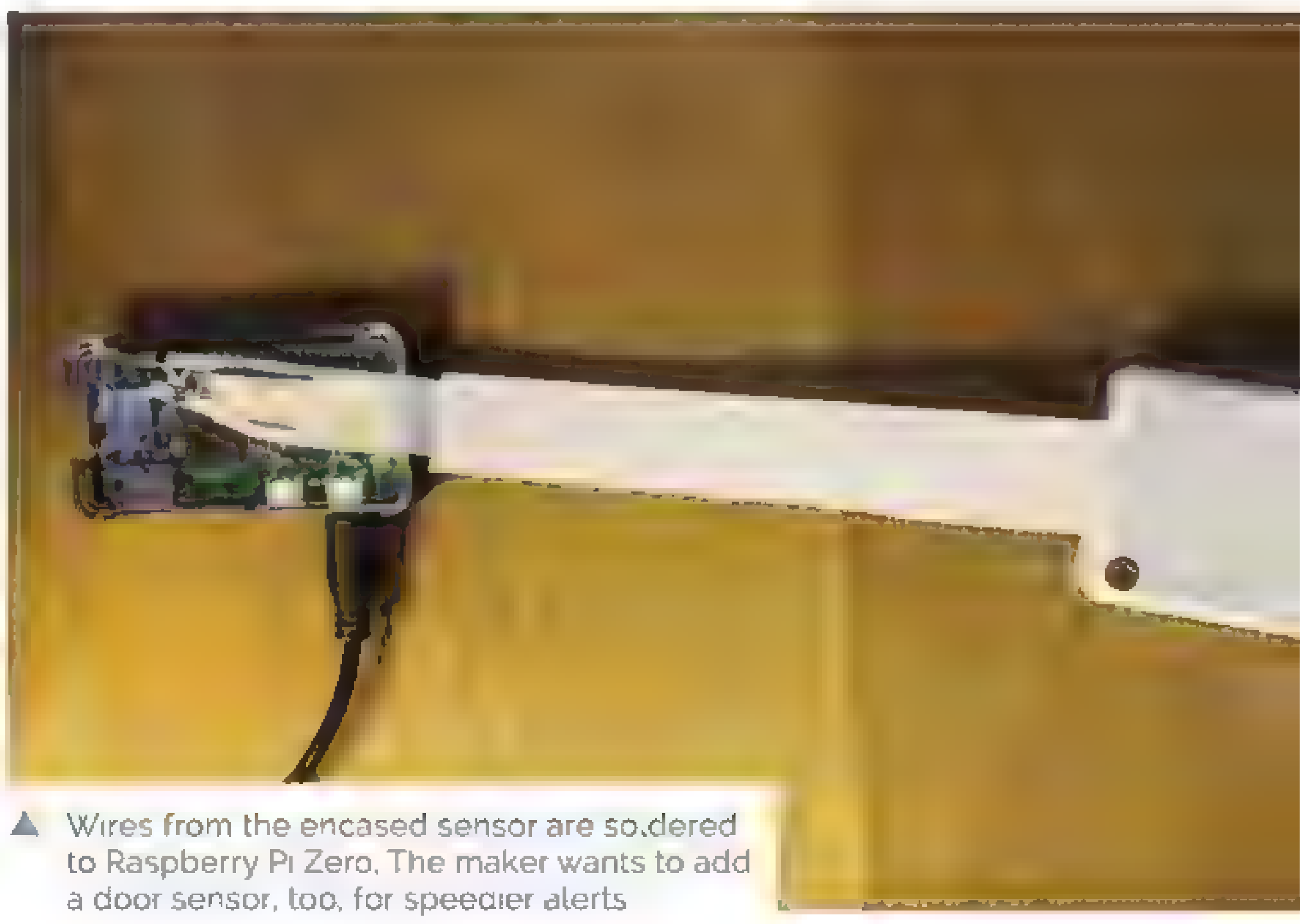
Rick got to work after witnessing the devastating effects a power cut can cause when his wife lost months' worth of breast milk during an outage. By connecting an Adafruit integrated temperature and humidity sensor to a Raspberry Pi Zero using a flat flex cable, he's developed a near-perfect, low-cost solution.

Keeping cool

"Raspberry Pi Zero is inexpensive and extremely capable," he says. "It's especially nice for small headless projects with wireless LAN connectivity.

> Raspberry Pi Zero is inexpensive and extremely capable

But the biggest reason I turned to this computer is because of the community, documentation, and tutorials that already exist. Unknown issues can be solved quickly by learning from the shared knowledge the community constantly generates."



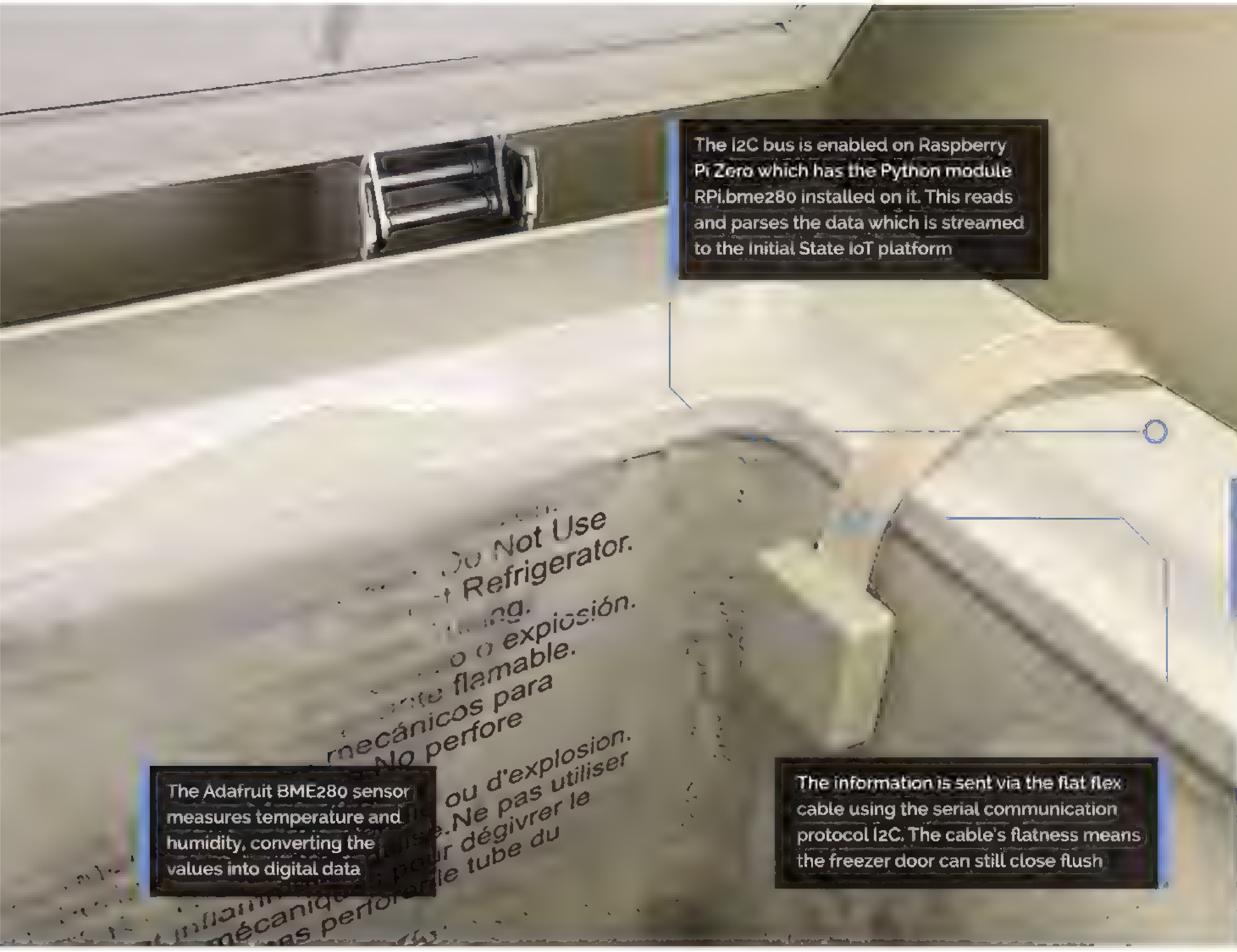
▲ Wires from the encased sensor are soldered to Raspberry Pi Zero. The maker wants to add a door sensor, too, for speedier alerts

At first, Rick played around with wireless sensors and items from the SmartThings platform. "It seemed elegant but the well-below-freezing temperatures totally degraded the batteries, and I wanted a reliable solution that didn't require constant fidgeting to keep it working."

He experimented with different wiring across a freezer's vacuum seal and he considered ultra-thin thermocouple wire. "Ultimately, though, I wanted to use the Adafruit sensor because of its temperature range and simplicity of integration with Raspberry Pi. The flat flex cable was key to getting signals across the freezer seal without leakage."

Avoiding the cut

Setting up the hardware proved straightforward. The relative difficulty came in creating the accompanying software. "Raspberry Pi OS Lite is great for headless applications, but it takes more



FACTS

- The whole setup costs around $30
- It uses Initial State
- The app boots when Raspberry Pi Zero starts up
- SMS and emails are sent if temperatures drop
- Mobile and desktop dashboards have been created

configuration to set up and install the modules, libraries, and SDKs properly," Rick says. "Getting wireless LAN setup headlessly, getting SSL access, and starting the app on bootup is not very discoverable either."

But once he had everything in place, coding the app was smooth. Rick used the IoT platform Initial State, so Raspberry Pi Zero only needs to stream the data via Initial State's Python SDK. "Without any extra coding on the device, I was able to create SMS/email triggers for temperature thresholds and data dropouts," he says. "I also calculated and displayed the various statistics and unit conversions using the expression engine." You can view the status of the fridge online (**magpi.cc/freezermonitor**).

The device has been running flawlessly for six months, already sending alerts via text and email when the power was cut due to storms around his home in Nashville. It has also let Rick know when the door was left open for too long, heading off some potential meaty disasters. "I'm now able to sleep easy knowing my prime briskets are safe and sound," he laughs.

▲ A dashboard has been created in Initial State. It receives data every ten minutes and allows for data to be viewed in real-time

Modern Jukebox

This 3D-printed mini jukebox combines a retro feel with modern music and podcast streaming. **Nicola King** has a dime ready

Bob Murphy

Bob (aka thisoldgeek) was a database administrator for the Clorox Company. In retirement, he has time to indulge his childhood dreams of making things

magpi.cc/thisoldgeek

There's something comforting and charming about the elegant lines of a jukebox, and Bob Murphy's childhood memories are a key driver behind why he decided to recreate this particular piece of vintage nostalgia. "My father had a 1940s floor-standing Zenith radio that I enjoyed listening to as a child," he shares. "That kindled a long-term interest in jukeboxes and Art Deco. I set myself the challenge of building my own jukebox, both for the learning experience and the satisfaction of having a personalised object."

So, Bob set about his task – a time-consuming build, as the printing of the casing and parts for the jukebox took over 75 hours to complete, not accounting for the failed prints. "My printer is a Prusa i3 MK3S, an excellent and reliable printer," Bob tells us, "but it requires mechanics like the belts to be properly tuned and adjusted. Once the printer and 3D software are dialled in, it's stick-to-it. Major parts took from 4–12 hours to print, over about four months. 3D printing is iterative, and I'm a poor planner. A few long reprints were needed once the electronics were ready to be fitted, to correct for clearances and other problems."

Light fantastic!

So, what do the internal workings consist of? "Raspberry Pi is at the base of a stack of electronics," says Bob. On top of Raspberry Pi, which runs the Volumio music app, Bob has an IQaudio DigiAMP+ HAT to route music to two Dayton Audio RS100-8 full-range speakers, and above that is an Adafruit Perma-Proto board powered by 5 V from the IQaudio amp. "That board hosts a SparkFun Sound Detector through an Arduino Micro. The Arduino uses the Sound Detector output to power NeoPixel strings for sound-reactive lighting effects."

▲ Mounted on Raspberry Pi, the electronics include an Arduino Micro, SparkFun Sound Detector, and an IQaudio DigiAMP+ connected to two speakers

The Modern Jukebox can be controlled either by the touchscreen display or via a web page from a computer, tablet, or phone, courtesy of Volumio. The user can raise and lower volume, choose music, skip tracks, pause/play, and so on. "On the back, there's an on-off switch and a switch that controls the NeoPixel LEDs," Bob reveals. "The 'Tube' LEDs around the outside can be set to sound-reactive, bubble/chaser through a colour wheel, or off. The LEDs around the grille rotate through a colour wheel at startup, then stay lit always."

Adopt, adapt, improve

Bob is very appreciative of the work of others that has inspired and aided him in his jukebox design. "This jukebox was such a long project, it wouldn't be possible without prior art from other makers," he notes. "I built on the efforts of Marco Gregorio

The vintage-style jukebox case is constructed from numerous 3D-printed parts

Connected to a Raspberry Pi running Volumio, the touchscreen can be used to control music playback and volume

Providing a sound-reactive light show, the 'neon' tubes are fitted with NeoPixel strips

FACTS

- The jukebox uses a standard 3A/12V DC output power supply
- Bob has made several interesting Raspberry Pi projects before…
- …including this aesthetically pleasing clock: **magpi.cc/artdecoclock**
- …and this vintage-style but technically modern radio **magpi.cc/miniz**
- Bob is planning to use Raspberry Pi Pico soon in a new project

> It takes an unknown amount of time and it's done when it's good!

(**magpi.cc/jukeboxdesign**) for the 3D jukebox model, and Michael Bartlett (**magpi.cc/ledmusicviz**) for the lighting effects."

Bob learnt from the work of these makers and adapted it for his own project. "I modified Marco's excellent full-sized, solid 3D Wurlitzer model. There were a very large number of components to review – what to keep, what is missing – to make this a scaled-down, modern-tech, 3D-printable object."

It's fair to say that Bob's Modern Jukebox has been a labour of love, but he's received some super-enthusiastic feedback from makers and admirers, with one friend asking, "Wow – is he taking orders?" Bob estimates that he has spent around 160 hours in the course of a year on his creation, but a quality output requires an investment of time. "I'm not on a deadline, so my approach to projects is like 'painting the Sistine Chapel' – it takes an unknown amount of time and it's done when it's good!"

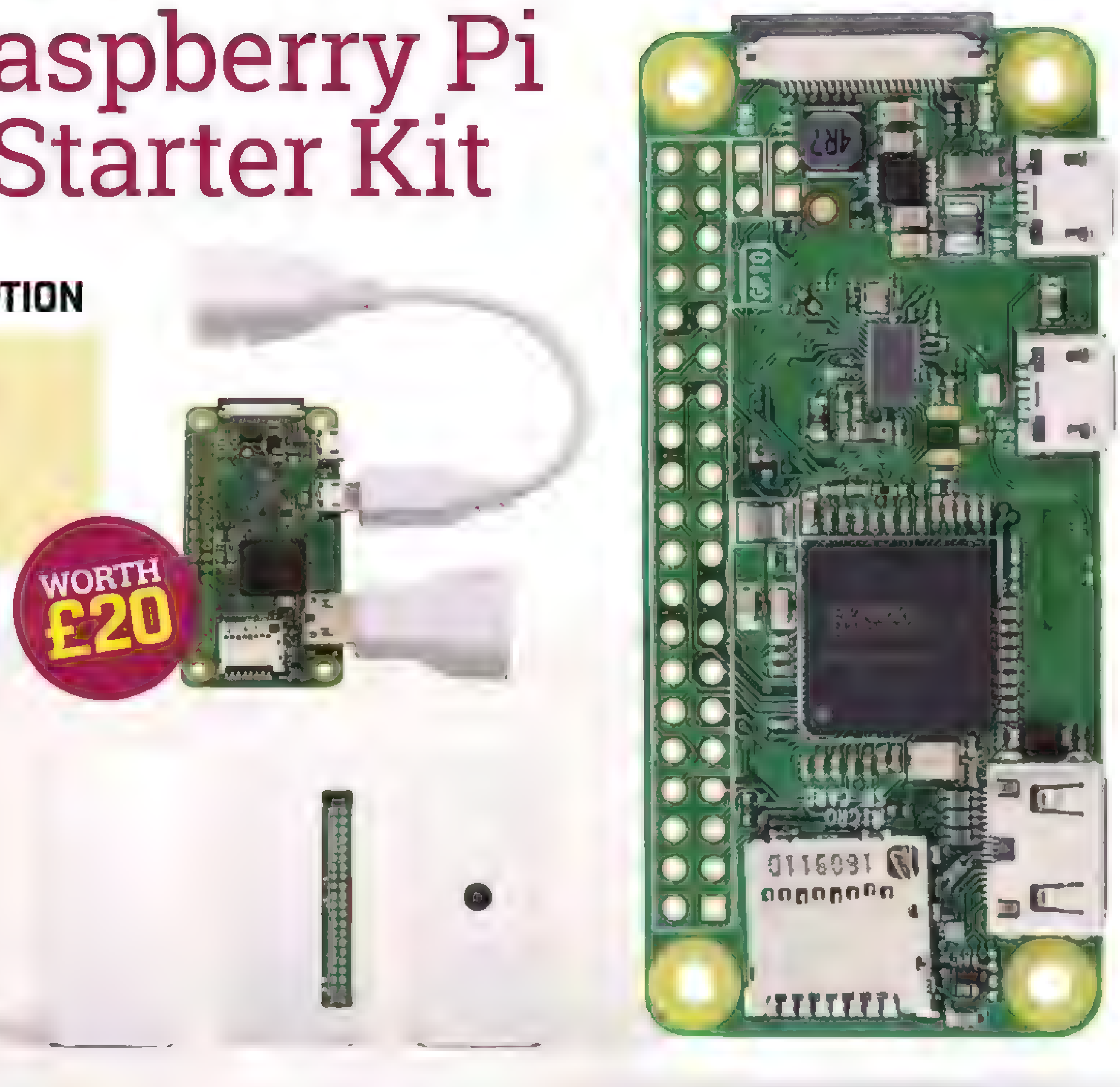
▲ Bob selects a song on the touchscreen panel, which shows Volumio's intuitive user interface

JOIN FOR 12 MONTHS AND GET A

FREE Raspberry Pi Zero W Starter Kit

WITH YOUR FIRST 12-MONTH SUBSCRIPTION

Subscribe in print today and you'll receive all this:



- Raspberry Pi Zero W
- Raspberry Pi Zero W case with three covers
- USB and HDMI converter cables
- Camera Module connector

This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time

Buy now: magpi.cc/subscribe

SUBSCRIBE on app stores

From £2.29

Available on the App Store

GET IT ON Google Play

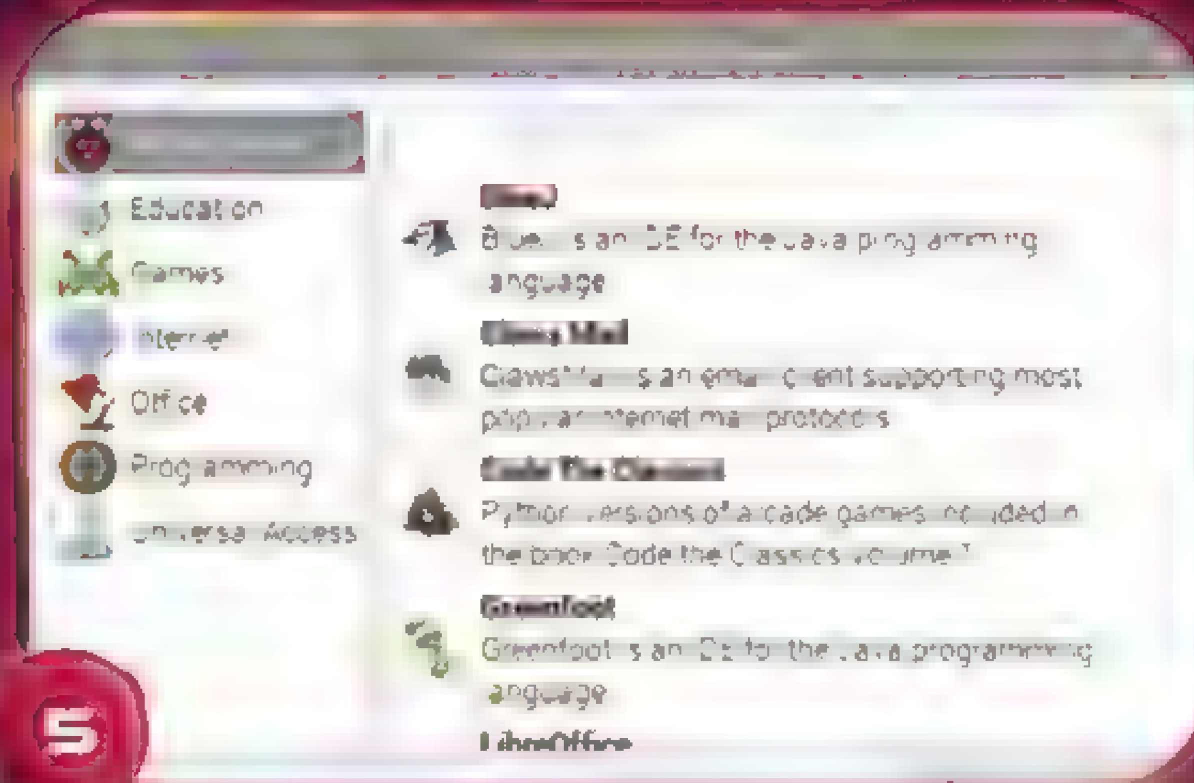
50 HACKS & HINTS

OUT OF THIS WORLD TIPS AND TRICKS FOR THE WORLD'S GREATEST COMPUTER. BY PJ EVANS

The world of Raspberry Pi computing is filled with possibilities. Sometimes this can lead to a feeling of being overwhelmed: Where to start? Before you go launching into your great project, it's worth taking the time to get to know your little computer and its big possibilities.

A Raspberry Pi is more than a microcontroller, such as its new sibling the Raspberry Pi Pico. Raspberry Pi is a full-blown computer with a complex operating system. Here we've compiled some of our favourite hints, tips, and short cuts to help you navigate this new world.

RASPBERRY PI OS TIPS



1 Create a custom OS

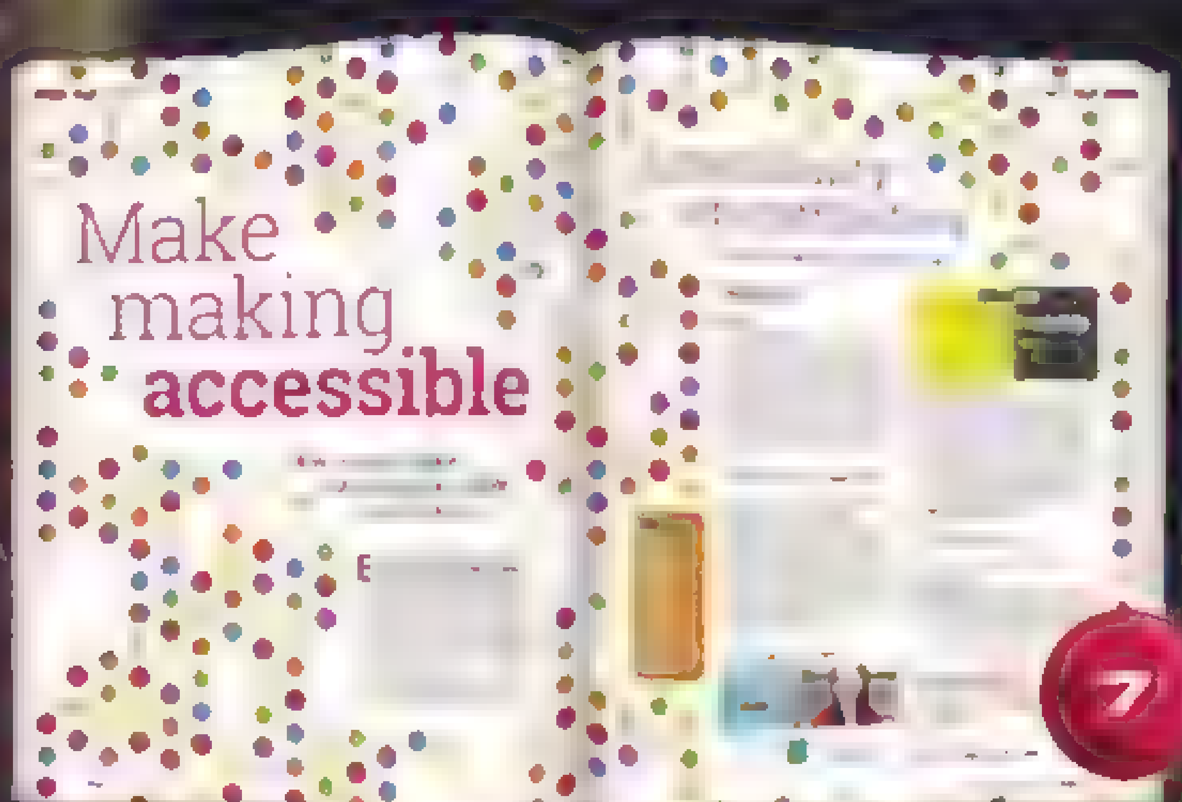
The Raspberry Pi Imager tool (**magpi.cc/imager**) makes preparing your SD cards a breeze, and the latest version now features a new Advanced menu. It is accessed by pressing **CTRL+SHIFT+X**. This enables you to preset your wireless LAN network, host name, and many other options. Ideal for multiple setups, trying new operating systems, and headless projects (where you don't need the graphical interface).

2 Go configure!

Raspberry Pi OS has features and settings that can be used to create an ideal setup. For example, if you want to interface with an I2C-based device or a Raspberry Pi camera, you'll need to enable support. The utility you'll need is Raspberry Pi Configuration. This can be run from the interface using Menu > Preferences > Raspberry Pi Configuration. You can also run it from the command line like so: `sudo raspi-config`.

3 Manage your memory

Raspberry Pi OS reserves a certain amount of memory for exclusive use by the GPU (graphics processing unit). If you're not making use of a desktop (such as a headless project) or if you are wanting a boost for a graphics-intensive application, you can change how much memory is allocated in Raspberry Pi Configuration under Performance.



4 Make your desktop your own

Not happy with the default look of the desktop? No problem. Visit Preferences > Appearance Settings and you can change fonts, colours, placements, and the background image. If you want to go further, you can find many guides to creating different environments on the web. You can even replace the default desktop window manager with alternatives.

5 Get more software

Raspberry Pi OS comes with lots of software to get you started, from programming environments to a complete suite of office software. This is just the beginning. Take a look at Menu > Preferences > Add New Software and you'll find thousands of packages that are one-click installs. Check Menu > Preferences > Recommended Software for the top picks.

6 Backup, backup, backup!

If you're storing data that you can't afford to lose, make sure to be backing up. One of the easiest methods is to clone the whole SD card (although it can get large!). Check out this guide: **magpi.cc/backup**.

7 Accessibility

Raspberry Pi OS has optional tools to assist people. In the Recommended Software application under Preferences, select the Universal Access category to install Orca, a popular screen reader, and Magnifier which enables easy screen zooming. Also, be sure to read our 'Make Making Accessible' feature in *The MagPi* issue 96 (**magpi.cc/96**).

8 Meet the terminal

On the top bar of your desktop, you'll see a small black box with a blue bar on top. This is your link to the command line via Terminal; the real power of Raspberry Pi OS. Pressing **CTRL+ALT+F1** to **F6** opens six different full-screen TTY (teletype) command-line interfaces. Press **CTRL+ALT+F7** to return to the regular desktop interface).

The Terminal gives command-line access to your system. When following projects in *The MagPi* and online, getting to know this way of talking to your Raspberry Pi is essential to growing your skills. Learn more at **magpi.cc/terminal**.

9 Pick your favourites

See that toolbar at the top of the screen? It is endlessly customisable. You can select your favourite apps for one-click access, change its size, position, make it disappear when you 'mouse away', even have multiple panels. Just right-click on the toolbar to explore all the options.

10 Take a screenshot

Need to take a quick picture of your desktop? Couldn't be easier. Just press **PRINT SCREEN** and a PNG image will be placed in your home folder. If you would like more options, such as a timer, the software does the work. The screenshot command `scrot` can be run from the command line, or you can install GNOME Screenshots. Read **magpi.cc/screenshot** for a full guide.

MAKERS' TIPS

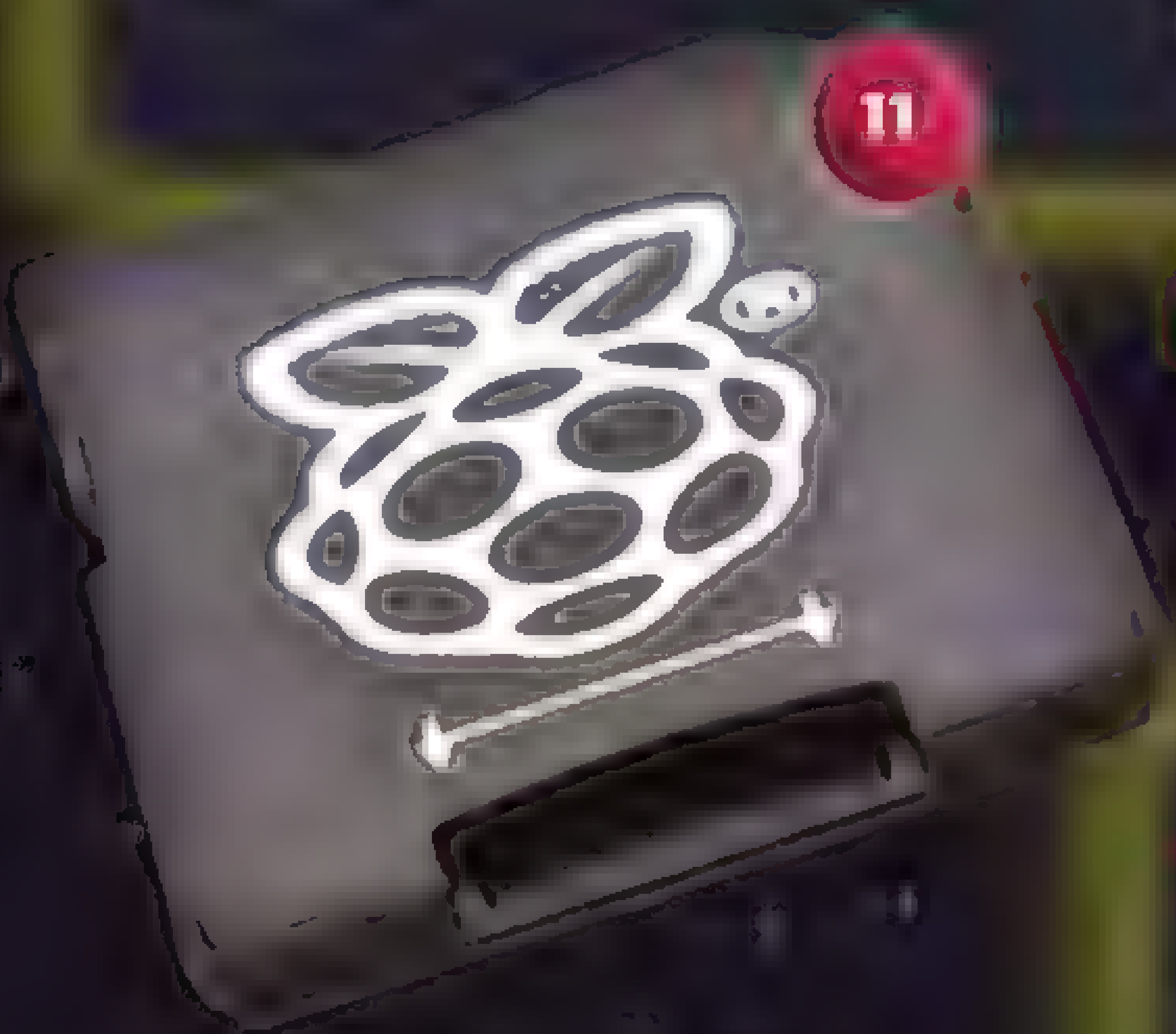
11 All the microSD Cards

One of the great advantages of using SD cards is their similarity to cartridges. Trying out a new OS or project build is as simple as swapping out that little piece of plastic. Make sure you buy cards from a reputable source: there are many out there that are not built for the workload to which an operating system subjects them.

12 Get kitted up

For newcomers to physical computing, the number of components available can be overwhelming. Check the popular Raspberry Pi market sites for brilliant starter kits with all you need and nothing more. Both the JAM HAT (**magpi.cc/jamhat**) and CamJam EduKit (**magpi.cc/edukit**) are great places to start. Take a look at the Starter Kits section in our Gadget Guide (**magpi.cc/gadgetguide**) for more equipment.

12



11

> "Raspberry Pi OS has features and settings that can be used to create an ideal setup"

13 Volt or bolt

Raspberry Pi GPIO pins can power accessories at 3V3 or 5 V. Check and check again before connecting devices to one of these. The wrong one may damage your device or the computer itself. Also, be mindful of how much current you'll draw. Finally, never connect a 5 V pin to a 3V3 pin: that would be the end of your Raspberry Pi.

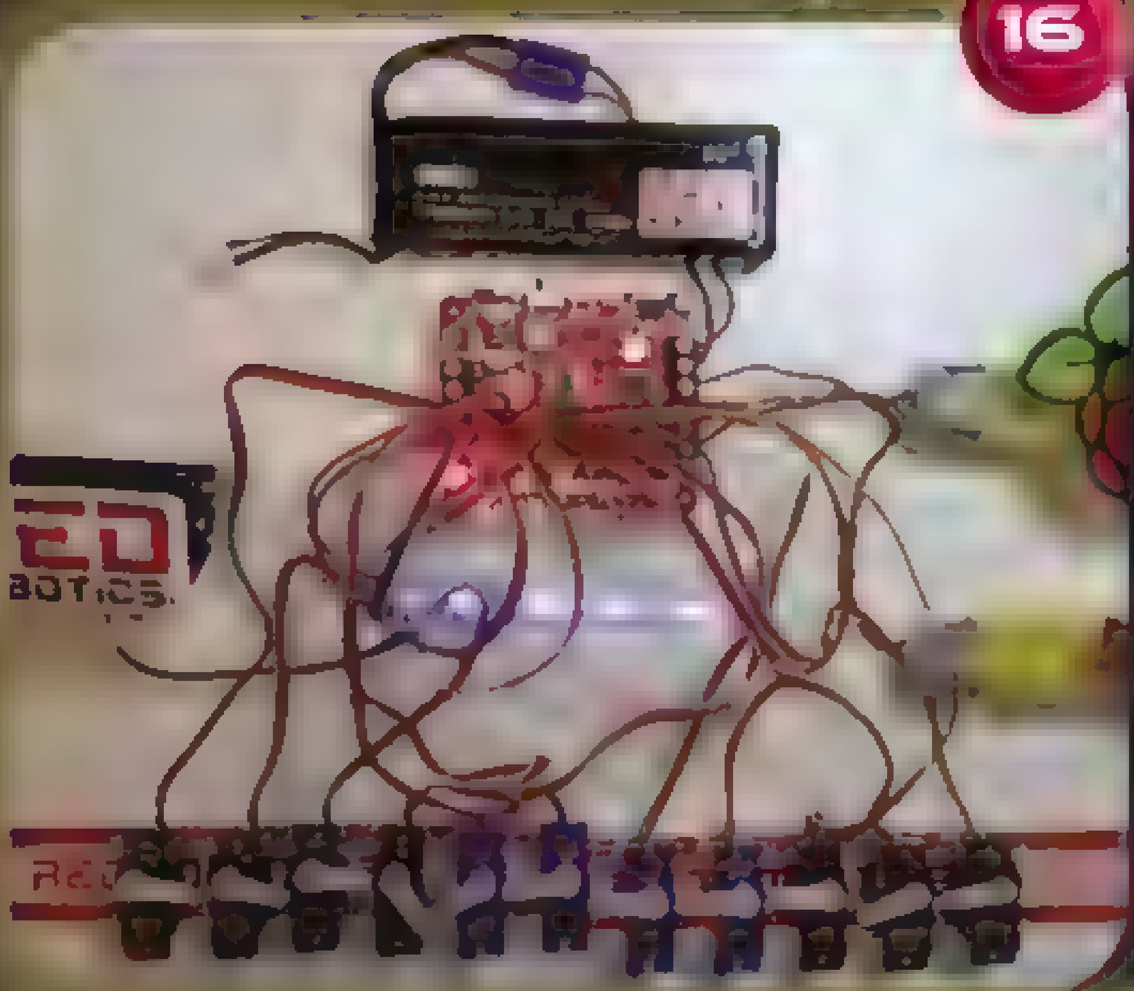
13

14 Build your own robot

The ability to drive motors and servos along with the horsepower required to process sensor data make it a natural choice. There's a lot to it though, and you can quickly find yourself with a full-time hobby. Robotics expert Danny Staple wrote a great guide to building a low-cost wheeled robot for us (**magpi.cc/robot**).

15 Go analogue

With the exception of the newest family member, Raspberry Pi Pico, the range of Raspberry Pi computers has been a strictly digital-only GPIO platform. If you want to read or write analogue data, such as light or moisture, you're going to need some help. Luckily many HATs can add this feature, such as Pimoroni's Automation HAT Mini (**magpi.cc/automationhatmini**).



16 Get your motor running

The current and voltage required for driving robot motors is not something that is going to keep your Raspberry Pi healthy for long. A robotics HAT is essential if you want your Raspberry Pi to control servos and motors. And to become truly portable, you'll need extra power too. For exceptional power and control, have a look at our RedBoard+ review in *The MagPi* issue 91 (**magpi.cc/91**).

17 Raspberry Pi, Pico and Arduino, oh my!

If you need analogue inputs and fast responses, consider using a microcontroller like Raspberry Pi Pico (**magpi.cc/pico**) in your project. Microcontrollers are much faster to respond to input. You can then use interfaces like UART, I2C, or SPI to feed data to your Raspberry Pi computer for more advanced processing.

18 Amazing camera work

You're probably familiar with the Raspberry Pi camera family. Did you know that they can do much more than take photographs? Use open-source tools like TensorFlow and OpenCV to add machine learning, object recognition, and even facial identification with simple code. There's an amazing resource at **pyimagesearch.com**

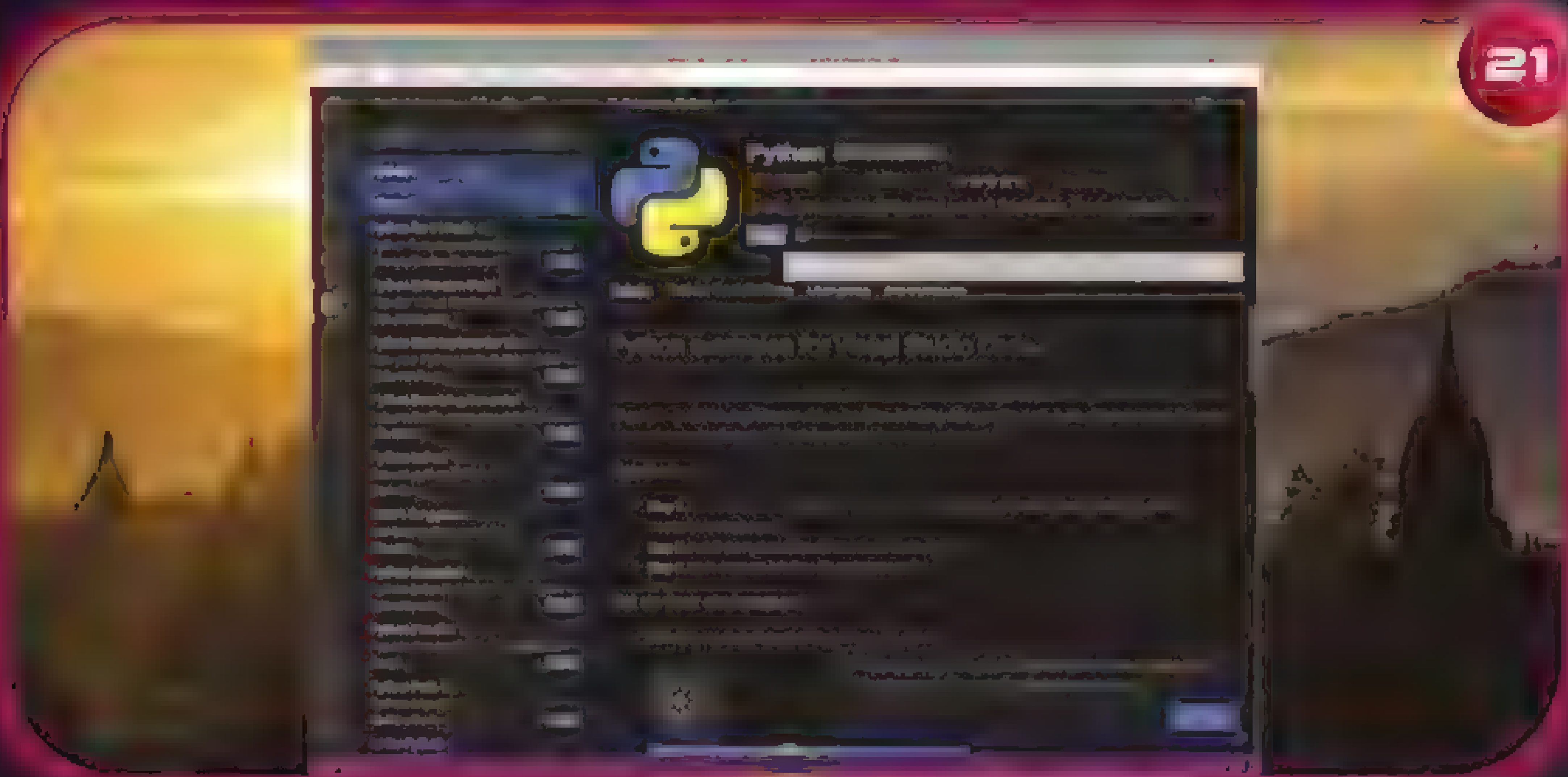
19 Get a 3D printing octopus

If you're a fan of 3D printing, you may have come across OctoPrint. This full suite of tools for controlling, monitoring, and recording 3D prints is also available as a custom Raspberry Pi image (**magpi.cc/octopi**). It's one of the best upgrades to a 3D printer you can get. Take a look at our 3D printing and making feature in *The MagPi* #97 (**magpi.cc/97**).

20 Right one for the job

You don't always need the full capabilities of Raspberry Pi 4. Consider a Raspberry Zero W for smaller projects. It may not have the performance of its larger sibling, but still offers the full Raspberry Pi OS, wireless LAN, and Bluetooth for an amazing price. Of course, the newest family member, Raspberry Pi Pico, brings amazing performance for under £5.

CODING TIPS



21 Supercharge your coding experience

Over the past few years, Microsoft's Visual Studio Code (**magpi.cc/vscode**) has become a de facto standard in development. It's more a turbocharged text editor than a full IDE, but its dizzying range of extensions brings it close. Visual Code is available for Raspberry Pi OS and also has extensions for remote editing over SSH and uploading to Raspberry Pi Pico.

> "No-code is a way of building applications without writing text"

22 Protect your code with version control

Git, the rather oddly named application, is an essential tool for every developer (**github.com**). It is a bit like time-travel for your code, allowing you to 'snapshot' your project and then rewind if you make a mistake, delete a file, or just change your mind. It's also great for collaborating on projects, enabling developers to work on the same piece of code without overwriting each other's work.

23 Code without coding

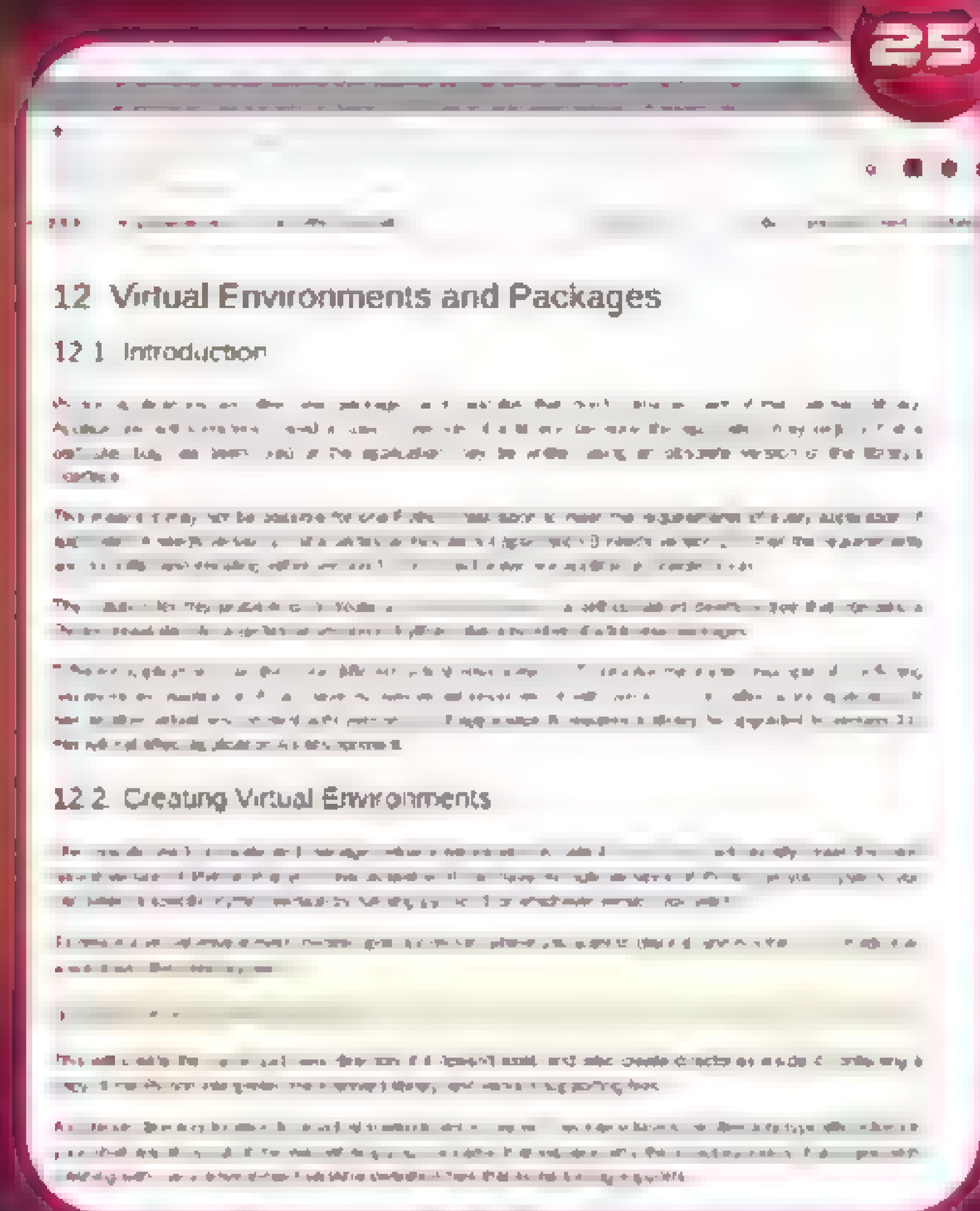
Tipped to be the way of the future, 'no-code' is a way of building applications without writing text, normally using a flowchart metaphor to get inputs, apply logic, and produce outputs. Raspberry Pi OS is ahead of the curve: try Scratch (**magpi.cc/scratch**), a building-block approach to making fun applications, or the more powerful Node-RED (**magpi.cc/nodered**).

24 Stuck? Ask Stack Exchange

The Stack Exchange family of websites is a Q&A service for developers and other technical roles. If you're stuck on a problem, the chances are someone else was too and turned to Stack Exchange for help. This is an amazing resource whose contribution to the coding community cannot be underestimated. There's a dedicated Raspberry Pi forum too (**magpi.cc/raspberrypisxc**).

25 Avoid code spaghetti

Programming languages like Python and Node.js rely on massive repositories of code libraries to extend their capabilities. Different dependencies can cause problems with conflicting versions. Virtual environments allow you to 'ring fence' a project and its dependencies so you can avoid this. Python 3 has this capability baked in and it's worth learning about it before coming unstuck. See **magpi.cc/venv**.



SECURITY TIPS

26 Change your password

It's obvious, but it bears repeating. Always, always change the default password. It takes under a minute and can save you significant grief, especially if your cool new project is connected in any way to the internet. Just run `passwd` from the command prompt or choose Menu > Preferences > Raspberry Pi Configuration and click Change Password.

27 Access Raspberry Pi from anywhere

The PiVPN project (**pivpn.io**) has recently added support for WireGuard, a new way of securely accessing your network from the internet. It only takes a few minutes to set up and clients are available for all major operating systems, as well as iOS and Android. An easy and secure way to get onto your home network from anywhere in the world.

28 Keep ahead of the bad people

Things move fast in the world of network security. Any Raspberry Pi device must be kept up-to-date with the latest software updates to ensure that any newly discovered vulnerabilities are mitigated. Regularly, run this command from the Terminal:

```
sudo apt -y update && sudo apt -y
full-upgrade
```

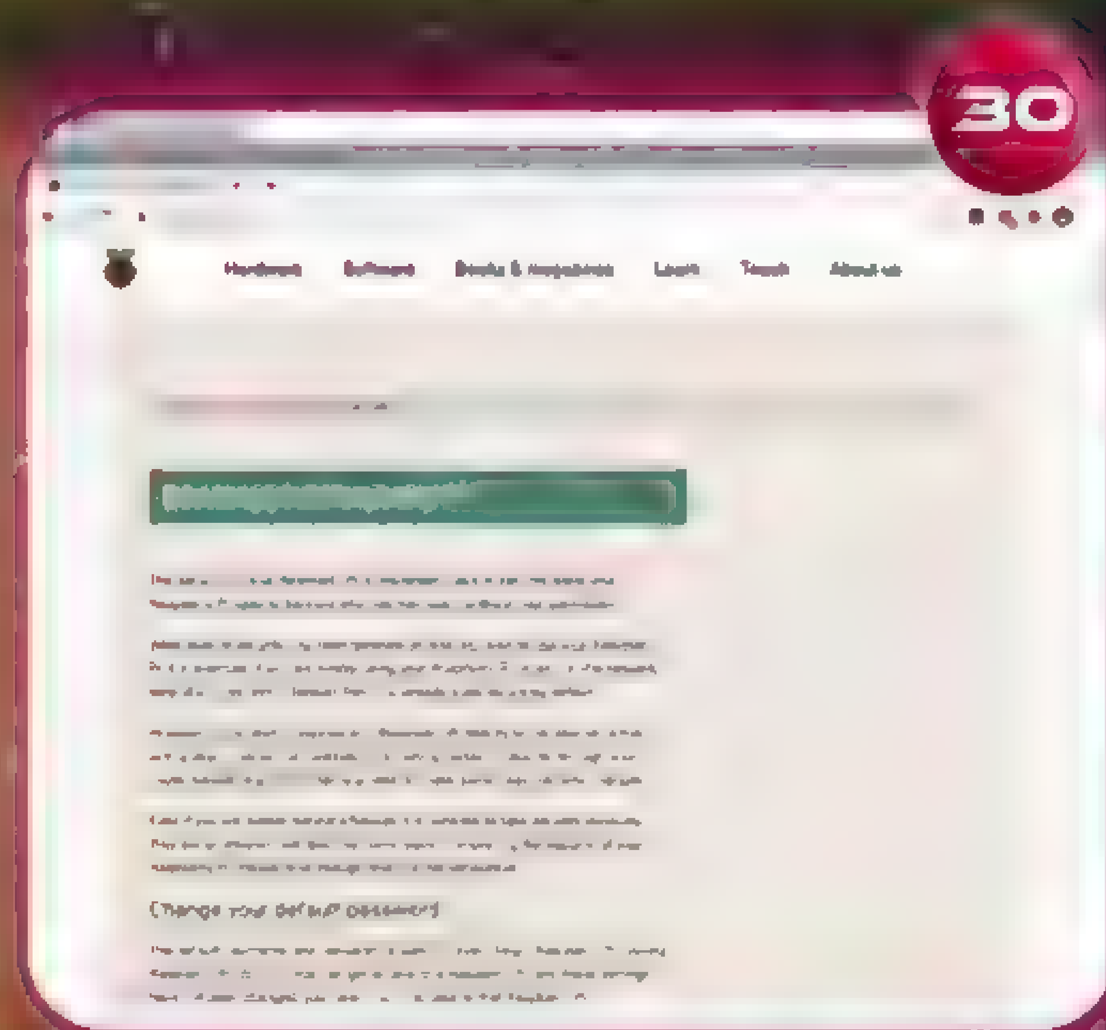
A full upgrade is better than just 'upgrade' because it will remove packages if needed to update the system.

29 Protect your privacy online

Fed up with ads? Don't want your every move being tracked? Pi-hole is the answer (**magpi.cc/pihole**): a DNS-based ad-blocking solution for your whole network. Installation is a breeze, there's a dedicated disk image for Raspberry Pi, and it can be configured to protect every internet-capable device on your home network.

30 Raise your digital drawbridge

If you're using Raspberry Pi OS in a potentially hostile environment (e.g. anything that isn't your home network), consider adding a software firewall such as UFW (Uncomplicated Firewall), which is based on the popular iptables software. This makes securing network access in and out of your device quick and easy. See 'Securing your Raspberry Pi' (**magpi.cc/security**).



NETWORKING TIPS

31 It's good to share

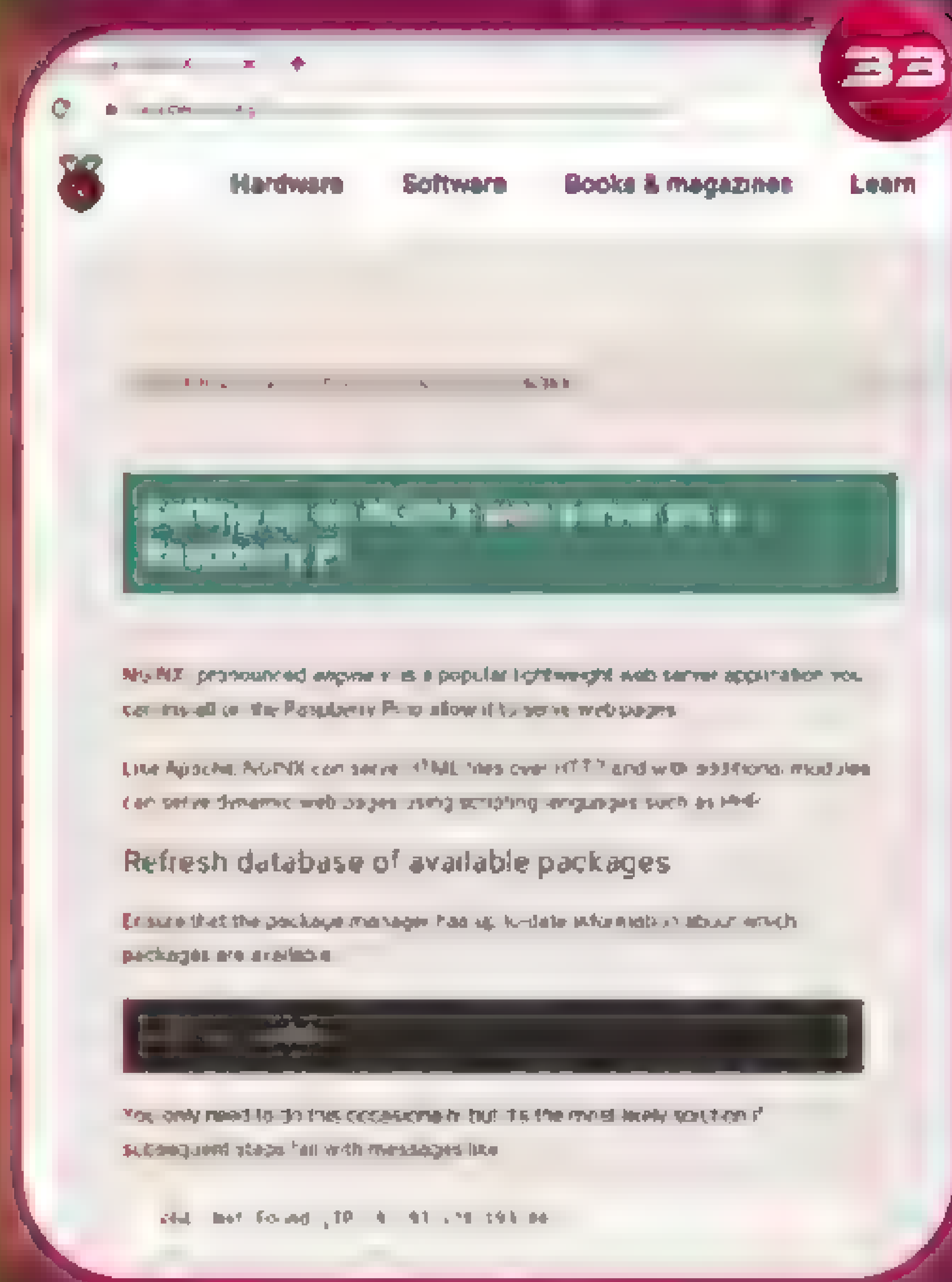
Raspberry Pi 4 makes for a brilliant NAS (network-attached storage) device. Use it in combination with a USB drive as a network-wide file repository. The popular Samba file-sharing software allows Linux, macOS, and Windows machines to map drives. Ideal for family file-sharing. See **magpi.cc/samba** for a tutorial.

32 What's in a name?

Every Raspberry Pi OS installation comes configured with a network 'host name' of 'raspberrypi'. When connected to your network, more than one of these is going to cause confusion, so it's essential to set the host name of the device to something unique in Preferences > Raspberry Pi Configuration.

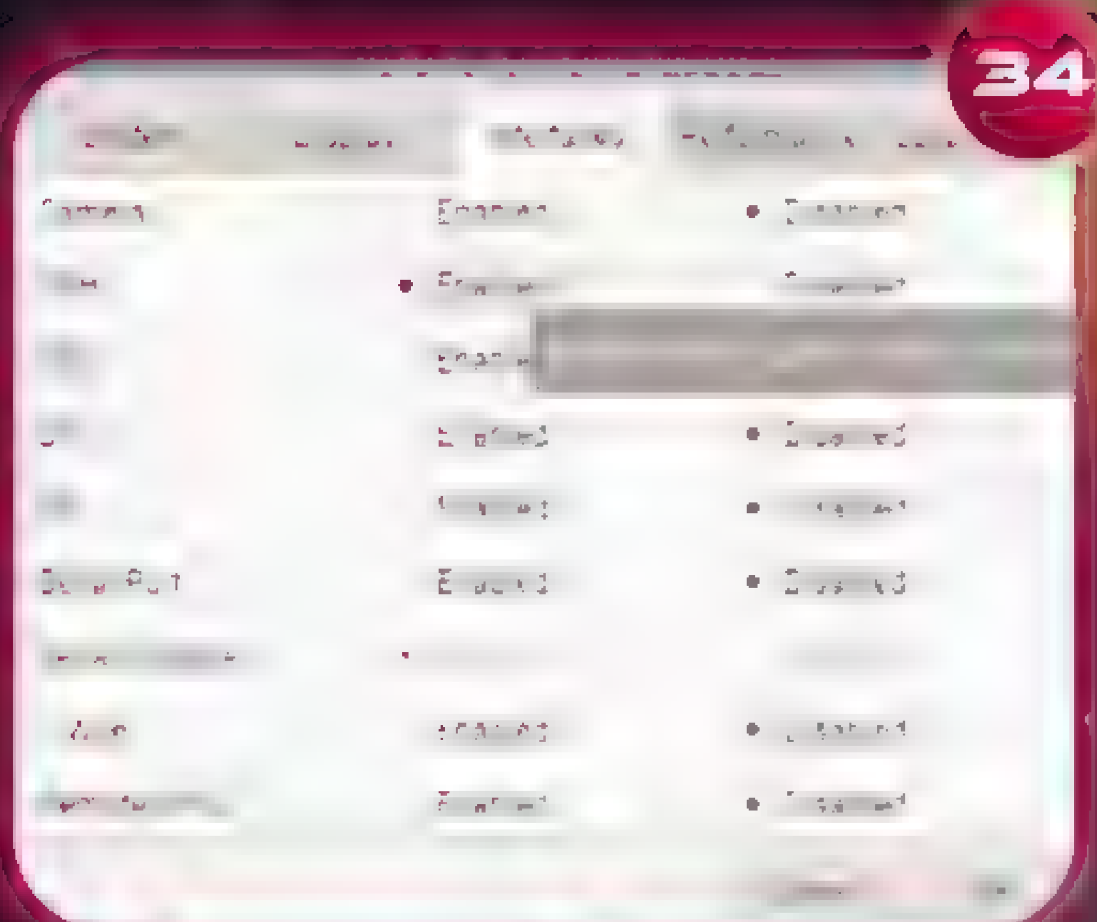
33 Build a web server

Raspberry Pi makes a brilliant web server and setting it up is as simple as running `sudo apt install nginx`. NGINX (pronounced 'Engine-X') is a modern, fast web server. Straight after install, you should be able to put your Raspberry Pi's address in a web browser and see a 'hello' message. See 'Setting up an NGINX web server' (**magpi.cc/nginx**).



34 Command Raspberry Pi from any computer

SSH or 'Secure Shell' is a common method of accessing a remote command line. It's a great way for controlling Raspberry Pi devices that are hard to reach or running without a monitor or keyboard ('headless'). Enable SSH in Raspberry Pi Configuration (under the Interfaces tab) and then log in from another computer using the ssh command-line application – or, if on Windows, puTTY. See 'Remote control your Raspberry Pi' (**magpi.cc/ssh**) for a tutorial.



35 I can VNC you

VNC is like SSH but for the entire desktop. It streams the desktop image to a remote computer and sends key presses and mouse movements back. Every Raspberry Pi OS Desktop has VNC available, but is not enabled by default, so just enable it in Raspberry Pi Configuration. Click the Interfaces tab and you'll find it just under SSH. You then need a VNC client app on your remote computer, such as VNC Viewer or TightVNC. See **magpi.cc/vnc** for a tutorial.

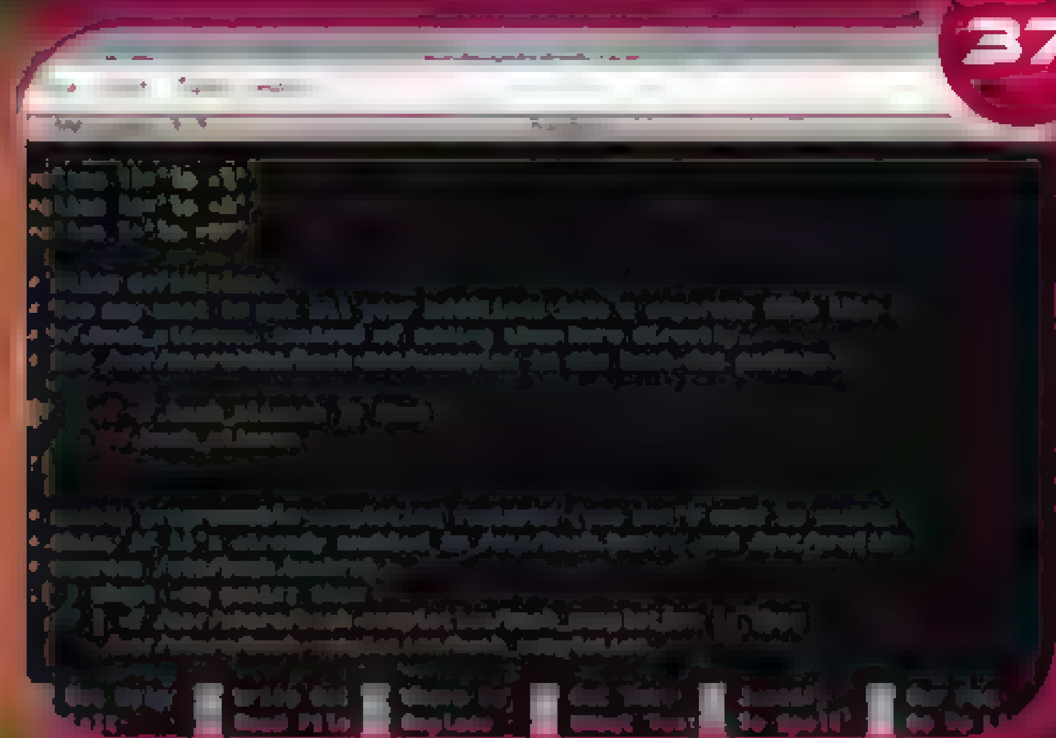
COMMAND LINE TIPS

36 The sudo that you do

Raspberry Pi OS is very secure. One of the ways it protects itself is to only allow system-wide changes to be made by the administration (or 'root') user. A simple way to run any command as the root user is to prefix it with 'sudo' (Super-user Do). See 'Root user/sudo' (**magpi.cc/sudo**) for more info on sudo and running commands as a root user.

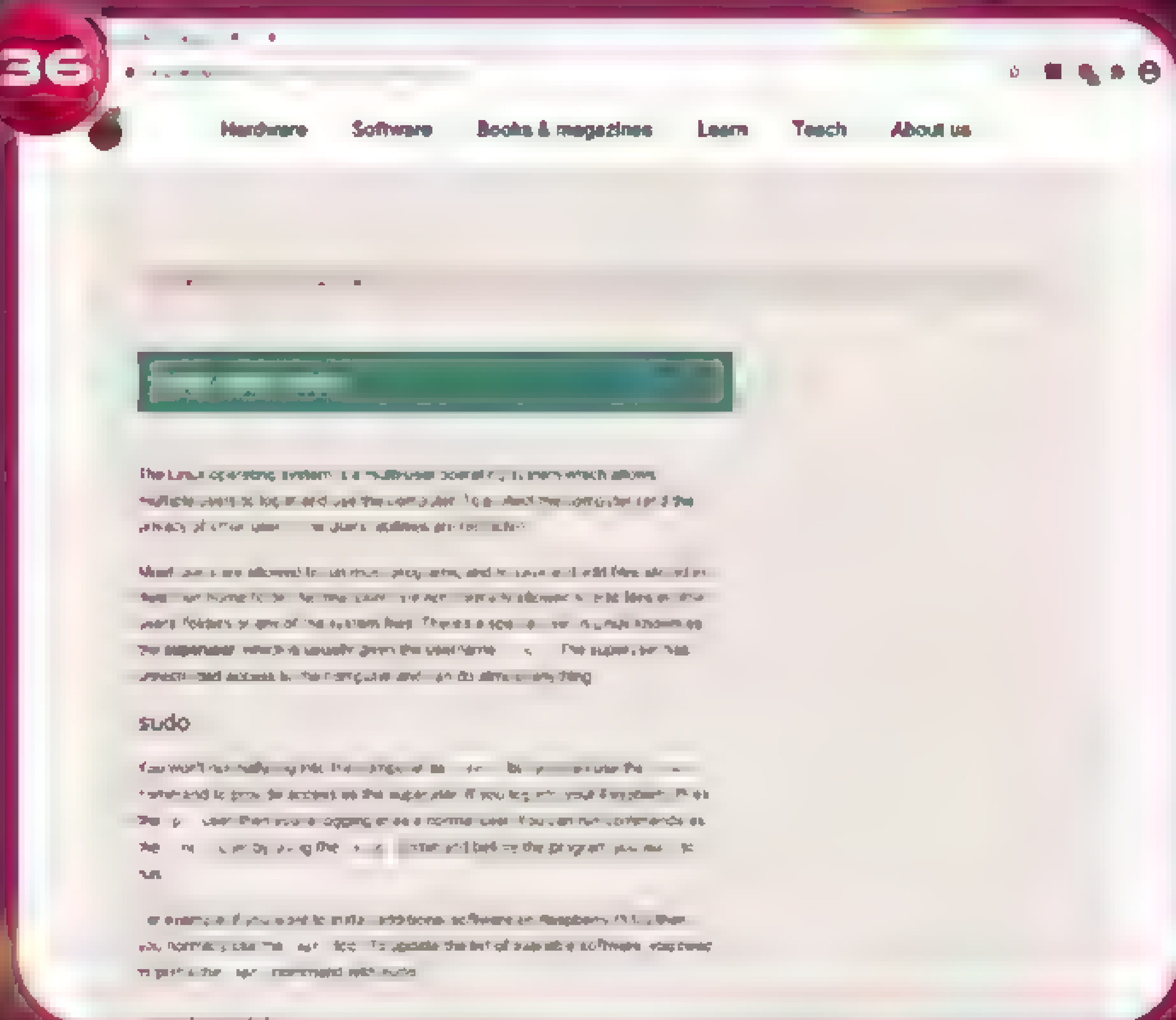
37 Save time with aliases

Find yourself typing the same commands over and over again? Create an alias! These are short cuts that you can define to convert long commands into anything you want. Create or edit the file **~/.bashrc**, then add them in like this: `alias l='ls --color=auto'`. A full guide can be found at **magpi.cc/bashrc**.



38 Take the command line back in time

Now how did that super-long Docker command go? Can't remember? Don't worry. Just type `history` and all your commands of the past are shown. You can also use the up arrow to cycle through your previous commands. Combine history with grep to find a certain line, e.g. `history | grep docker`. You can also press **CTRL+R** to do a reverse search of your command-line history. Just enter any word (such as 'apt' to search for the last command you entered with that word).

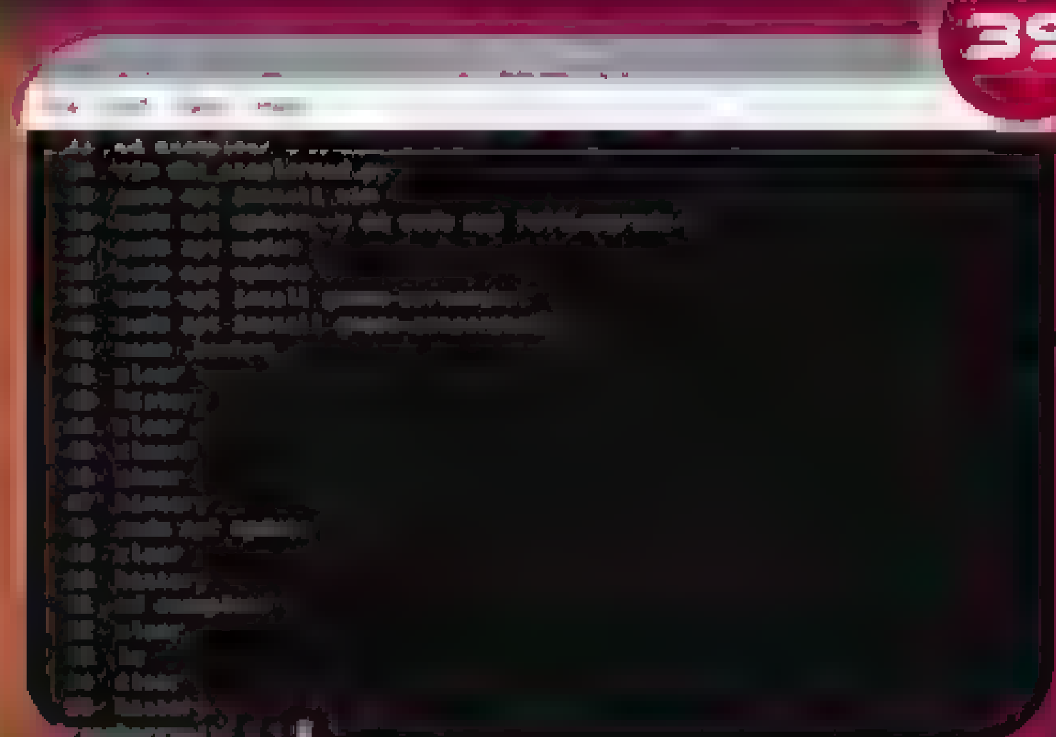


39 !!

No, that isn't a typo. Bang-bang (or !!) means 'the last command you ran'. Why is this useful? Because of those times you forgot to prefix a command with sudo. Don't retype the whole thing, just enter `sudo !!` and you're set. Forget to pipe a command? Try `!! | <next command>`. You can also combine `!` with `history` to run commands. Just use `!` with the number in history.

40 What's going on?

If your Raspberry Pi is running hot or seems to be slow, it may be that a process has gone rogue and is spiralling out of control. There are a few ways to find out what the culprit is. The easiest is to run Task Manager from Accessories. If you don't have a desktop, `ps ax` will list every process running and `top` will show processes in order of CPU usage.



SUPER SHORT CUTS

41 Super-fast terminal

Raspberry Pi OS desktop comes with a range of useful keyboard short cuts to help you get the job done quickly. Need a terminal in a hurry? **CTRL+ALT+T** will have your command line up in a second.

42 Arrange your windows

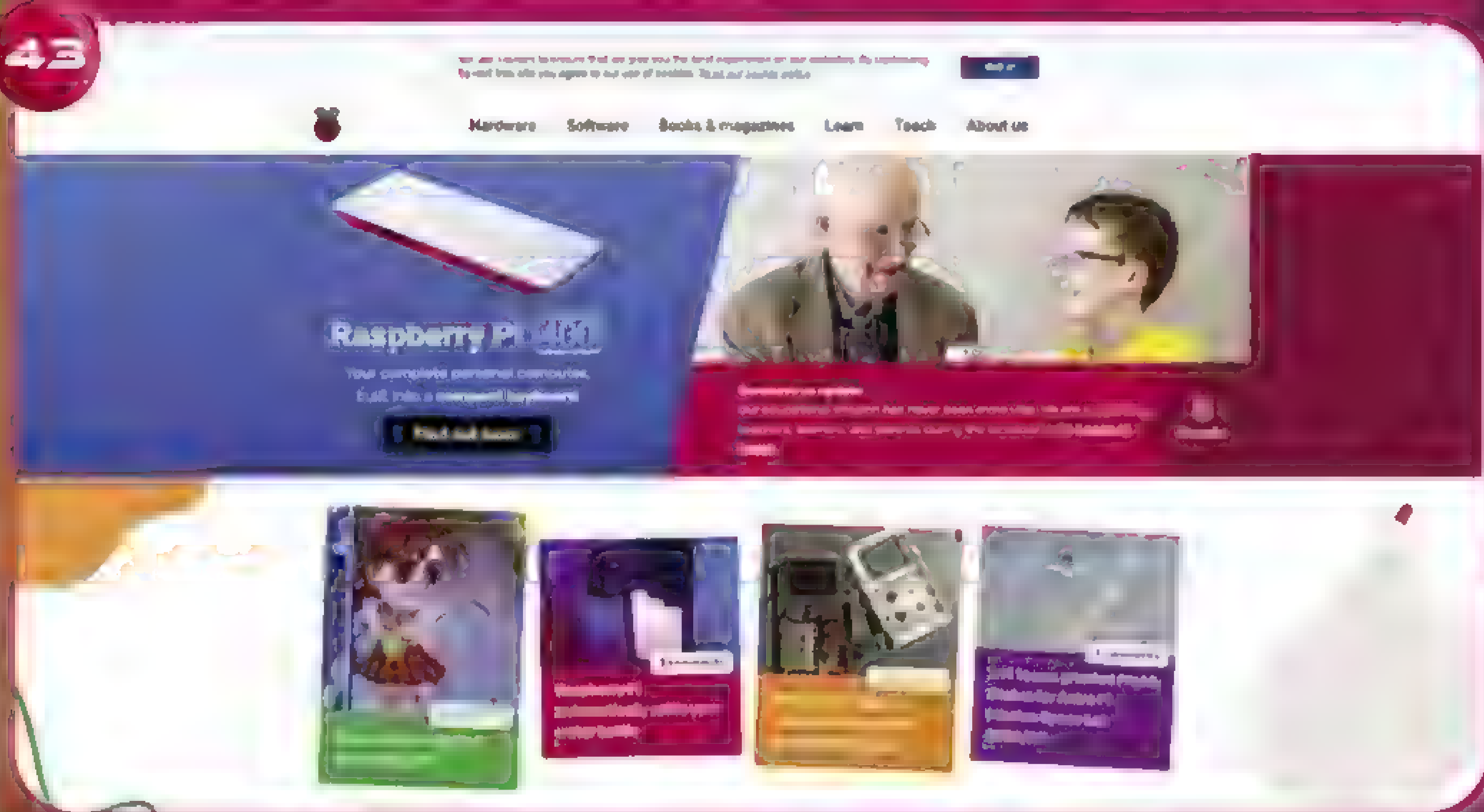
Desktop looking a little crowded? There are a few short cuts that can help you get your windows in alignment. With a window focused, use **CTRL+ALT+UP ARROW** to set it full screen. **CTRL+ALT+Left** or **Right Arrow** will move it to half of the desktop.

43 Take up all the screen!

If you're working on a particular window and maybe you're on a smaller screen, such as a Raspberry Pi touchscreen, you might find the menu bar is taking up useful space. Simply pressing **ALT+F11** with your chosen window focused will expand it to fill all the available screen. **ALT+F11** again will reduce it down to its original size.

44 Fast running

Need to access the menu? Don't waste time and energy with that mouse – just click the command key on your keyboard! That's the Raspberry Pi logo on the official keyboard, or the Windows logo or Apple button on their respective keyboards. If you want to run a command quickly, just press **ALT+F2** to bring up a 'run dialog'.



45 ALT and click

Did you know you can drag windows around by holding down the **ALT** key and dragging any part of the window with the left mouse button? It's handy for when a window ends up off of the side of the screen and you can't access the menu bar. Also, you can resize a window by holding down the **ALT** key and clicking and dragging with the right mouse button. Much easier than finding the corner of a window.

46 You're on mute!

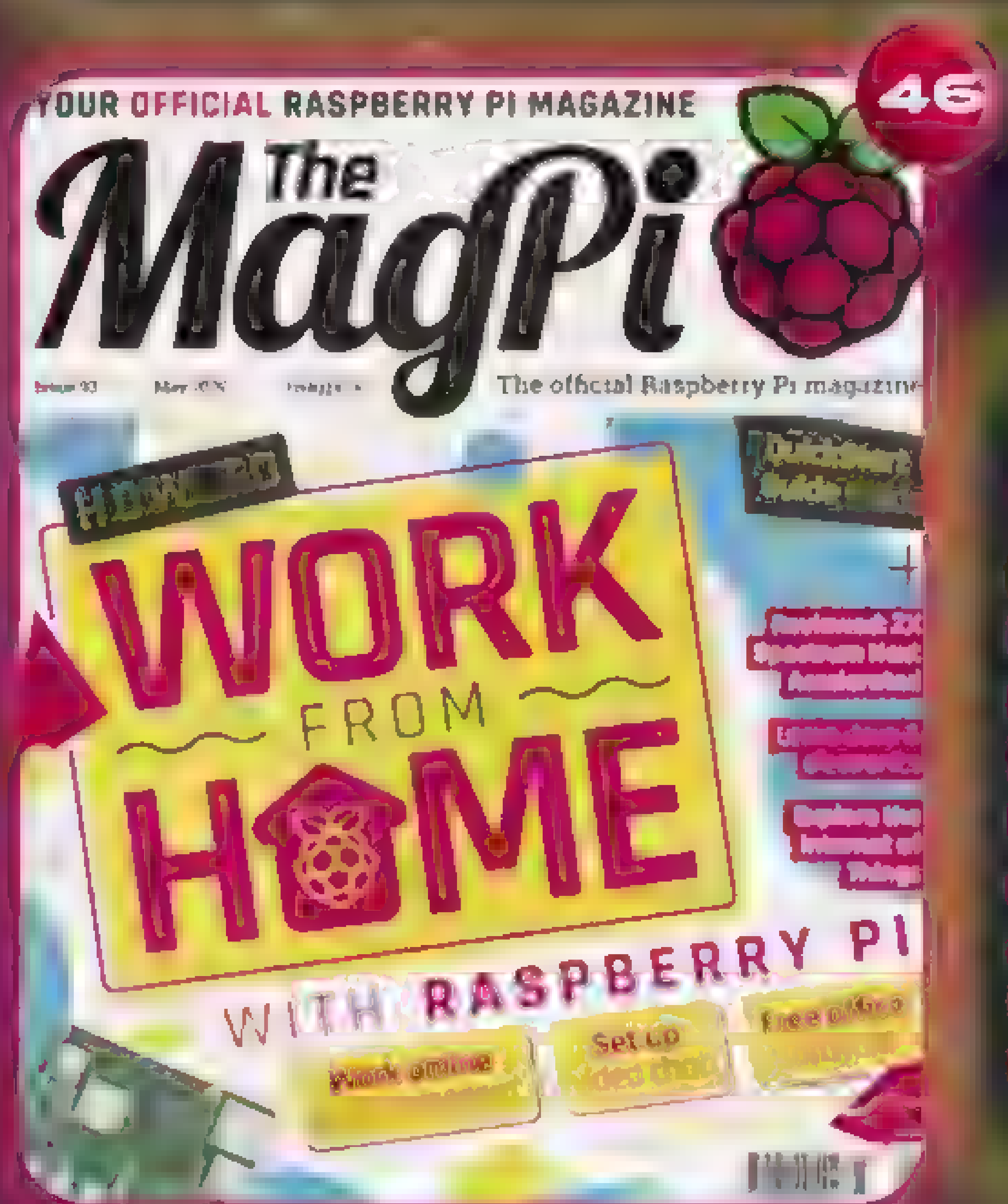
Yes, the application of 2020 can be coaxed into running on a Raspberry Pi. Zoom has a semi-hidden web app that will work with Chromium. When going to a meeting, click the link and an error will appear. Dismiss this and click 'Click here' until it says 'start from your browser'. It's a demanding application, so we recommend a Raspberry Pi 4 for this one. See 'Working from home with Raspberry Pi' (**magpi.cc/workfromhome**) or check out *The MagPi* issue #93 (**magpi.cc/93**).

47 Printing

Printing can be a tricky thing to get working. Luckily, Raspberry Pi supports two key technologies to get things working. Firstly, CUPS is an Apple-driven, open-source project to bring printing to Linux-based systems. Secondly, IPP (Internet Printing Protocol) is a 'common ground' that most modern printers support if you find ARM-based drivers are unavailable. Read 'Printing at home from your Raspberry Pi' (**magpi.cc/printing**).

48 Dashboards

It's perfectly possible to use Raspberry Pi 4 as a 'daily driver' workstation. However, if you're using something a bit more powerful, why not set up a Raspberry Pi dashboard? Use open-source projects such as Grafana to create charts, alerts, and more. Combine with a Raspberry Pi touchscreen and case for a beautiful sidekick to your PC. See *The MagPi* #98 for a guide to building a portable Raspberry Pi with a touchscreen.



49 Home automation

The excellent Home Assistant (available as an SD card image for Raspberry Pi) allows web-based control of a huge range of internet-connected devices. Perfect for creating the right lighting and temperature for your home office. If you're concerned about air quality, combine it with a Pimoroni Enviro+ for particle monitoring. See 'Home of the future' in *The MagPi* issue 104 (**magpi.cc/104**).

50 LibreOffice

The LibreOffice suite has come on in leaps and bounds over the past few years. This has become a serious contender in the world of office applications. Word processing, spreadsheets, databases, presentations, and more for free. The entire collection is installed with a 'Recommended Software' Raspberry Pi OS image or within a few clicks on the desktop.

Build an arcade machine: Assemble your cabinet

Once your arcade cabinet kit arrives, it's time to put everything together

K.G. Orphanides

K.G. is a writer, maker of odd games, and software preservation enthusiast. Their family fully supports the idea of an arcade machine in the living room.

In this tutorial, we will assemble an arcade cabinet, fit controls, and mount a monitor. You should follow the video or written assembly instructions for the model you buy, but we'll go through the process so you know what to expect and how to handle the awkward bits.

Kits don't necessarily come with the screws and bolts you'll need to attach parts such as speakers, speaker grilles, and monitors, so check that you have all the hardware you'll need before you start.

Our cabinet is an Omniretro Bartop Arcade King with a stand (**magpi.cc/kingbartop**), made of 16 mm black melamine laminate, and we are using a 24-inch monitor.

01 Lay out your parts

MDF and melamine laminate are light, cheap, and sturdy when assembled, but can be susceptible to damage if dropped or pivoted hard on an edge or corner.

Make some space and put down towels to protect the cabinet parts and your floor from one another. If your unit consists of a separate bartop and stand, build them one at a time. Read or watch the manufacturer's instructions and make sure that you have all parts, fixings, and tools to hand before you start.

02 Preparation

Assembly varies from brand to brand. If access to the assembled cabinet is restrictive, you may have to fit your buttons and joystick to it before you put it together

Similarly, attach speakers to the inside of the marquee bottom and speaker grilles to the outside before you assemble the cabinet. If you're working with laminate, mark up the screw positions through their holes with a paint pen and use a 3 mm bit to drill pilot holes.

If you've already decided on your marquee, control panel and bezel graphics, your life will be easier if you apply these to their acrylic sheets before assembly (we'll be looking at this in detail in a later tutorial).

> Put down towels to protect the cabinet parts and your floor from one another

03 Assembly

If you're comfortable with self-assembly furniture, an arcade cabinet shouldn't present too much trouble, but a second person can be helpful for fitting and moving awkward parts.

Ours has a control panel with a hinged access door beneath it, so we attached this hinge first

It's a good idea to fit your speakers and grilles before assembling the cabinet, but it's possible, if fiddly, to do it afterwards

We have put U-moulding onto the edges of the cabinet now to protect them, loosely secured with standard double-sided tape at the ends. We'll re-secure this properly after decorating the cab

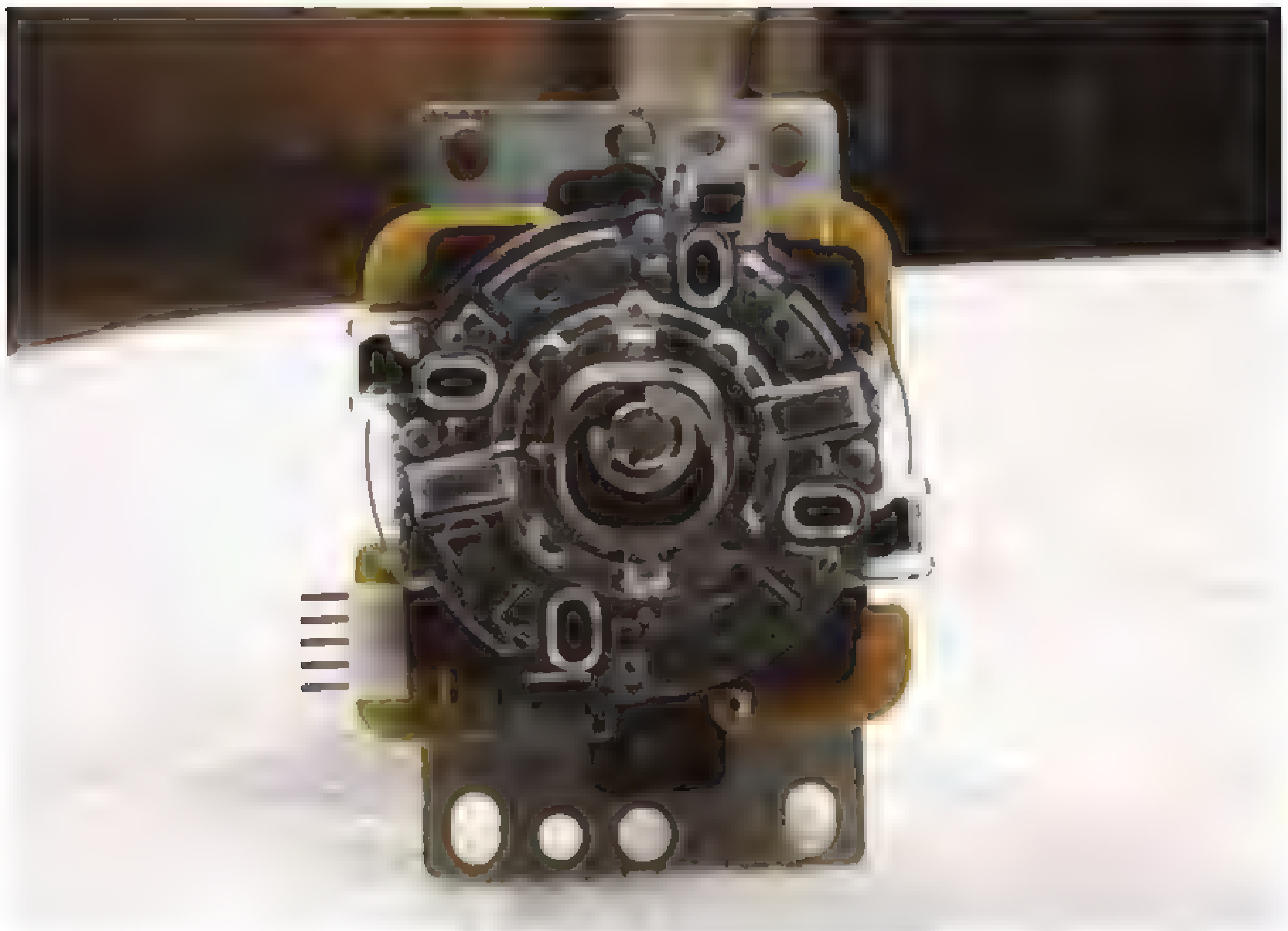
Top Tip

Snap-out

Snap-in buttons can be hard to remove without damage. The ButterCade Snap-out tool for push-buttons (**magpi.cc/snapout**) is a plastic device to help with this.

You'll Need

- Screwdrivers, spanners, Allen keys, crimping tool
- Cordless drill
- Drill bit set Screwdriver bits, drill bits, countersinks, tank cutters
- Additional bolts, screws, female spade connectors (to mount components)
- Dremel (recommended) and 3mm drill
- Paint pen (silver if you have black laminate, black for MDF)
- Old towels or sheets to protect parts
- Foam cleanser and microfibre cloths (to clean your cabinet and acrylics)



The underside of a Sanwa JLF-TP-8YT joystick. Note the e-clip securing the central shaft

then set the panel aside. We then attached the hinge for the bartop's rear access door and base, lined this part up with the cabinet's hood-like top, and bolted all of these parts to one side panel laid on top of them.

Lining bolts up with pre-drilled holes for this kind of build can be fiddly. If you have trouble, screw the bolts through the side panel until they're protruding, and use them to help find the correct positions.

04 The control panel

With one side now in place, slide in the control panel and bolt it to the same side as the other parts. Next, attach the marquee bottom that houses the speakers, which should already be mounted at this point.

With this model, we then close the latch on the rear access door and carefully flip the entire cabinet over onto the now-secure side panel. This is the best time to slide the marquee and screen acrylic panels into place. If you've not already applied graphics to them, leave their protective film on – it's easy to peel off later.

We now position the second side panel. We recommend again screwing in the bolts until they just protrude from the opposite side to help you lower the panel securely and accurately onto its pre-drilled holes.

05 Feel the power

Drill a hole at the back of your bartop and run the power bar's cable out through it to connect directly to a plug socket.

Some suppliers will wire a socket and bar for you, but note that international plug standards differ. Use a plug bar that can be surface-mounted inside of the cabinet.

While you're back there, cut a hole to accommodate a booted Ethernet cable or, more tidily, a screw-down Ethernet extension port. This will make Steam Link game streaming easier.

06 Extending your shaft

If your cabinet is over 16 mm thick, you'll want a longer than standard joystick shaft. Shafts are easy to swap, but watch out for parts dropping out.

Like most sticks, our Sanwa JLF-TP-8YT's shaft is held in place at the bottom by an e-clip. Hold the unit upside down, press on the bottom of the shaft with your thumb, and use a small flat-head screwdriver in your other hand to pull the clip off, using the slots in it. Pull the old shaft gently out from the top and push the new one in, carefully setting the pivot at the top and the spring and black plastic actuator at the bottom into place.

Use a thumbnail to depress the actuator and slide the e-clip back into place. You can also use pliers or your screwdriver to help push it on. For a demonstration, see this YouTube video on changing joystick shafts: **magpi.cc/joystickshafts**.

To fit the VESA mount, place the cabinet face-down, then put the mounted monitor face-down on the front acrylic screen. Use a tape measure to help with positioning

Warning!
Mains electricity & power tools

Be careful when handling projects with mains electricity. Insulate your cables and disconnect power before touching them. Also, be careful when using power tools during this build.

magpi.cc/drillsafety
magpi.cc/electricalsafety

Cable tidy

Cable lacing is a cable management technique where a nylon cord is used to bind wires together. It can be used to create incredibly neat builds, like this Arcade Stick by Gordon Hollingworth, Raspberry Pi's Chief Product Officer.

Gordon learnt to cable-tidy this way as an apprentice for the MOD. "Tying the knot has to be done in a very specific way to avoid it looking untidy," he tells us, "basically a capital offence in the apprenticeship!" Gordon's cables have knots regularly at 1 cm, which keeps them smart. "We learnt this way because when you put a box into a plane or tank with some equipment in it, the vibration will shake apart pretty much any connection in the first hour. So this was the way it was done when electronics was more about wires connecting things than PCBs."

You can buy nylon cord and learn more from RS Components (**magpi.cc/cablelacing**).

Snap-in buttons are held in place by plastic clips. Connect your DuPont GPIO cables first to make internal wiring easier

> There's room to slide the screw slots on most joystick mounting plates

07 Installing your joystick

Two plastic dust washers come with Sanwa joysticks. Slide one onto the shaft before you mount the stick onto the underside of your control panel.

When mounting your joystick, position it, mark up the position of the top right screw-hole on the joystick's baseplate with a paint pen, and drill a pilot hole, being careful not to go all the way through.

Attach your joystick by that screw, make sure it's centred, and mark up the next hole or holes. There's room to slide the screw slots on most joystick mounting plates, so you've got a bit of wiggle room when it comes to the final fit.

Don't worry too much about the orientation of your joystick – position it where it won't get in the way of the rest of your wiring. These are nominally designated up, down, left, and right positions; you can reassign these through wiring and in software.

Finally, slide the second dust washer onto the shaft on the other side and screw the joystick's ball on.

08 World of buttons

Snap-in buttons are ideal for thick wooden cabinets – plastic clips hold them in position inside the holes drilled for them. If you have an acrylic cover for your control panel, the buttons will hold it in place.

It's a good idea to attach your spade connectors to DuPont GPIO jumper cables before installing them, but you'll have to connect the shared ground cable after they're in place. We wired GPIO to the right and shared ground to the left connector on each button, but it doesn't matter which goes where.

Where you have longer stretches between buttons, skip a connector on the ground chain to give yourself some extra cable to play with.

You can label the end of each GPIO cable for later ease of connection to Raspberry Pi, but they're not too hard to trace in most cabinets.

Top Tip

Foot the bill

To help the cabinet stand on an uneven floor, you can fit four rubber feet to its underside.

▲ To make it easier to line up the sides of your cabinet with their pre-drilled recieving holes, partially screw in each bolt until a couple of millimetres protrude on the far side

> If you find that you now can't reach or fit a part, don't panic

▼ Before construction, lay out the parts of your bartop on some old towels to protect them

09 Bolt screen to VESA mount

Screen mounting can be fiddly. Most cabinets come with a baton-like wooden VESA mount that's designed to be screwed into place from the inside. Start by bolting your monitor to the mount. Unless you're working with a specialist cab designed for giant screens, you'll be using a 75 mm or 100 mm VESA mount. These usually take M4 bolts and have a depth of 10 mm. So if bolts aren't included, you'll need four, at a depth of 10 mm plus the depth of your mount, although you can get away with shorter if you countersink them.

10 Screw VESA mount to the cabinet

Place the bartop face-down on the ground, protecting it with a towel. Take the protective plastic off the acrylic on the inside of the cabinet. Clean the acrylic with a microfibre cloth and anti-static foam cleanser.

Lay the monitor, attached to the cabinet's VESA mount, face-down on the acrylic inside your cabinet. On the interior sides of the cabinet, mark up the position of the holes in the brackets on each edge of the VESA mount. Remove the mount, drill pilot holes, then replace and screw down the display and its mount.

If your monitor has a front power button, you can use adhesive chair leg floor protectors as soft spacers to stop it from being pressed by the acrylic screen.

11 Don't panic

If you miss a stage in your build and find that you now can't reach or fit a part, don't panic. Speakers – and any other components in need of securing – can be attached internally using strong double-sided foam tape.

Most external parts can be drilled and fitted in situ. If you want to deal with decoration last, then you can sometimes pop out your acrylic panels or, better, remove one side and reattach it.

As you'll see from the photos, we have temporarily applied U-moulding to protect the edges of the cabinet. U-moulding is easy to remove and refit or replace, assuming you don't glue it down, but T-moulding is a little harder to remove cleanly.

We're now ready to connect Raspberry Pi. That will be covered in the next tutorial.

Create GUIs with Python: Tic-tac-toe

Use your GUI to control a simple game

Laura Sach

Laura leads the A Level team at the Raspberry Pi Foundation, creating resources for students to learn about Computer Science

@CodeBoom

Martin O'Hanlon

Martin works in the learning team at the Raspberry Pi Foundation, where he creates online courses, projects, and learning resources.

@martinohanlon

Now that you have learnt how to make a basic GUI, let's add some more programming logic behind the scenes to make your GUI work as the means of controlling a game of tic-tac-toe (also known as noughts and crosses).

Create a new file with the following code:

```
# Imports ---------------
from guizero import App

# Functions -------------

# Variables -------------

# App -------------------
app = App("Tic tac toe")

app.display()
```

Create the board

Let's begin by creating the widgets which will make up the game board. A traditional tic-tac-toe board looks like the one shown in **Figure 1**.

You'll use buttons to represent each of the positions on the board, so that the player can click on one of the buttons indicating where they would like to move. To be able to lay out the buttons on a grid, let's create a new type of guizero widget called a Box.

A Box is a container widget. This means that it is used for containing other widgets and grouping them together. Add it to the imports at the top of your code:

```
from guizero import App, Box
```

Set the Box to have a grid layout and add it to your app – before the `app.display()` line, as with all widgets.

```
board = Box(app, layout="grid")
```

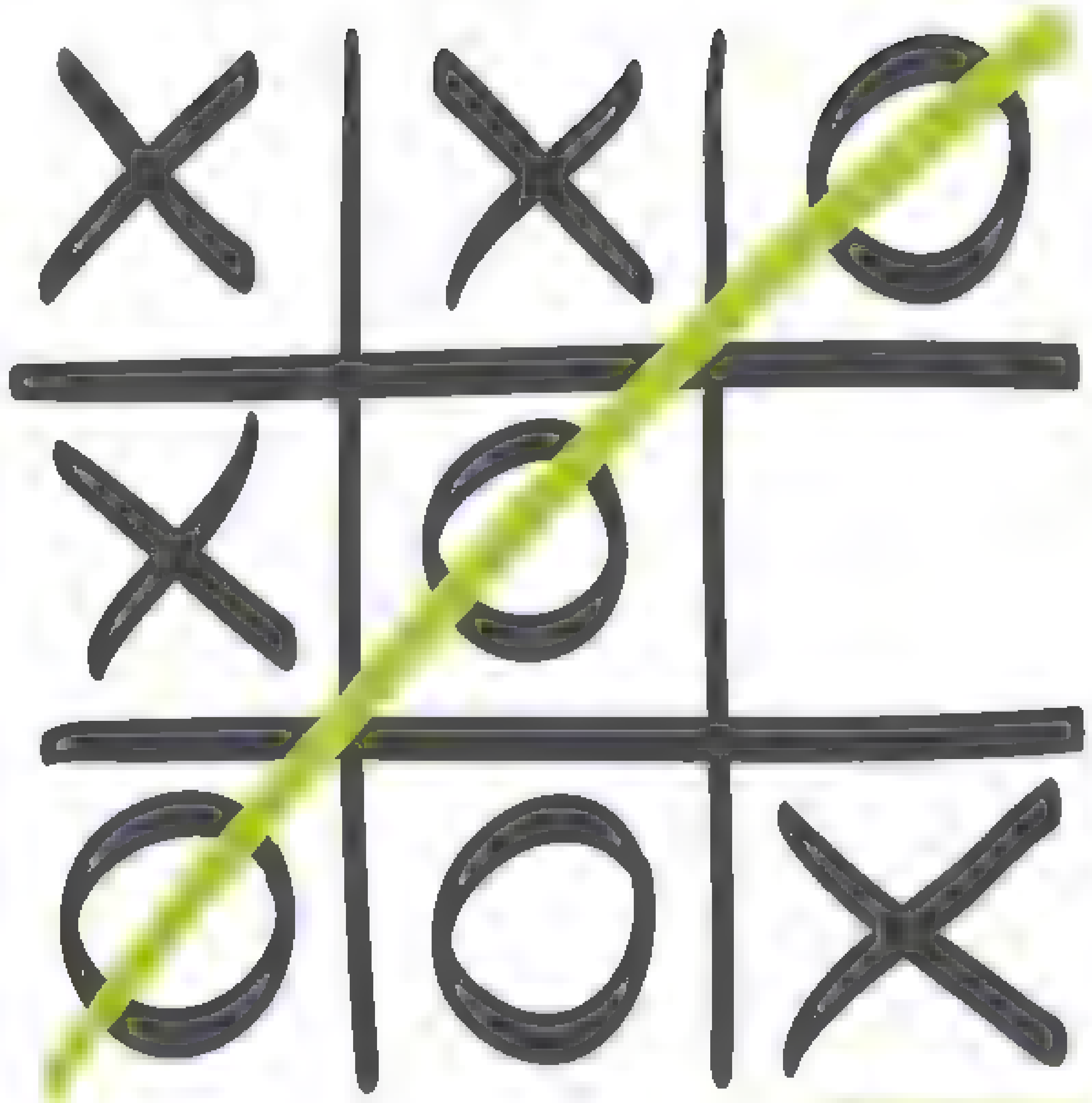


Figure 1

▲ **Figure 1** A typical game of tic-tac-toe

If you run your program at this point, you won't see anything on the screen because the Box itself is invisible.

Now let's create the buttons to go inside it. You will need nine buttons in total, so instead of creating them individually, you can use a nested loop to generate them all automatically and give them co-ordinates. First, add PushButton to your list of widgets to import and then add this code immediately after the code for the board you just created.

```
for x in range(3):
    for y in range(3):
        button = PushButton(
            board, text="", grid=[x, y],
width=3
        )
```

Notice that there are two loop variables: **x** from 0 to 2 and **y** from 0 to 2. As we iterate and generate

Figure 2 A grid of nine buttons to play tic-tac-toe

tictactoe1.py

> Language: **Python 3**

DOWNLOAD THE FULL CODE:
magpi.cc/guizerocode

```
# Imports ---------------
from guizero import App, Box, PushButton

# Functions -------------

# Variables -------------

# App -------------------
app = App("Tic tac toe")

board = Box(app, layout="grid")
for x in range(3):
        for y in range(3):
        button = PushButton(
board, text="", grid=[x, y], width=3)

app.display()
```

tictactoe2.py

> Language: **Python 3**

```
# Imports ---------------
from guizero import App, Box, PushButton

# Functions -------------
def clear_board():
    new_board = [[None, None, None], [None, None, None], [
None, None, None]]
    for x in range(3):
        for y in range(3):
            button = PushButton(
                board, text="", grid=[x, y], width=3)
            new_board[x][y] = button
    return new_board

# Variables -------------

# App -------------------
app = App("Tic tac toe")

board = Box(app, layout="grid")
board_squares = clear_board()

app.display()
```

buttons, each button will be added to the board, which is the Box container you created earlier. The button will be given the grid co-ordinates `x,y`, meaning that each button is neatly placed on a grid at a different position!

Your code should now look like **tictactoe1.py**. The result of running it is shown in **Figure 2**.

Underlying data structure

You might notice that when you create the buttons using a loop, you are creating nine buttons automatically and every single one is called `button`. How will you be able to refer to each of these buttons in the program?

The answer is that you need an underlying data structure to hold a reference to each button, and for this you will use a two-dimensional list.

Let's create a function which we can call to clear the board. It is a good idea to do this in a function so that you can reuse the code once the game has been played to reset the board and allow the player to begin a fresh game.

In the functions section, add a new function called `clear_board`.

```
def clear_board():
```

Your first job inside this function is to initialise the data structure for the board. Let's assume at this point you have not created any buttons, so you can initialise each position on the board as `None` – the element in the list now exists but does not yet have a value. Add the following line, indented, to your function.

```
new_board = [[None, None, None], [None,
None, None], [None, None, None]]
```

Next, move the nested loop code from your app section into the `clear_board` function. Make sure the indentation is correct.

Inside the inner (y) loop, add a line of code to store a reference to each button at its x,y co-ordinate position within the two-dimensional list so that you can refer to it later.

```
new_board[x][y] = button
```

Finally, after the loops end, return the `new_board` you have just created. Your function should look like this:

```
def clear_board():
    new_board = [[None, None, None],
```

tictactoe3.py

> Language: **Python 3**

```python
# Imports ---------------
from guizero import App, Box, PushButton, Text

# Functions -------------
def clear_board():
    new_board = [[None, None, None], [None, None, None], [None, None, None]]
    for x in range(3):
        for y in range(3):
            button = PushButton(board, text="", grid=[x, y], width=3, command=choose_square, args=[x,y])
            new_board[x][y] = button
    return new_board

def choose_square(x, y):
    board_squares[x][y].text = turn
    board_squares[x][y].disable()

# Variables -------------
turn = "X"

# App -------------------
app = App("Tic tac toe")

board = Box(app, layout="grid")
board_squares = clear_board()
message = Text(app, text="It is your turn, " + turn)

app.display()
```

```python
                    [None, None, None],
                    [None, None, None]]
    for x in range(3):
        for y in range(3):
            button = PushButton(
                board, text="", grid=[x,
y], width=3
            )
            new_board[x][y] = button
    return new_board
```

In the app section, initialise a list called `board_squares` and set it to call the new function you just created.

```python
board_squares = clear_board()
```

This variable will be assigned the value of the `new_board` you created within the function, which should be a blank board with nine buttons. Make sure that you create this variable after the code for creating the Box, otherwise you will be trying to add buttons to a container that does not yet exist.

Your code will now resemble **tictactoe2.py**. Save and run the program and you should see an identical result to the one you had at the end of the last step, but now you have a hidden two-dimensional list data structure to let you reference and manipulate the buttons.

If you want to see what your 2D list looks like, you could add a print command to print the `board_squares` list: `print(board_squares)`. You should then see nine lots of `[PushButton] object with text ""` appear in the shell.

Make the buttons work

At the moment, your buttons don't do anything when you press them. Let's make a function to attach to the button, so that when it is pressed, the button displays either X or O depending on which player chose it.

First, create a variable in the variables section to record whose turn it is. You can choose to start with either player, but we will choose to start with X.

```python
turn = "X"
```

This now means that you need to display on the GUI whose turn it is (**Figure 3**) so the players don't get confused. Add Text to your list of widgets to import:

```python
from guizero import App, Box, PushButton,
Text
```

Reset the game

At the start, you wrote a function called **`clear_board`**. This may have seemed unnecessary at the time, but in actual fact it was thinking ahead to when the game has ended. Since tic-tac-toe is quite a short game, it is likely that someone might want to play more than one game in a row.

Can you add a reset button to your game, which only appears once either someone has won the game, or the game was a draw? The button should call the **`clear_board`** function and reset the **`turn`** variable as well as the message reporting whose turn it is.

Hint: You will need to check the guizero documentation to find out how to hide and show widgets, so that your button is not visible all of the time during the game.

Hint: Create a new function which takes care of everything you need to do to reset the game, and call that function when the reset button is pressed. Don't forget that in your function you'll need to specify some variables as global.

tictactoe4.py

> Language: **Python 3**

```
# Imports ---------------
from guizero import App, Box, PushButton, Text

# Functions -------------
def clear_board():
    new_board = [[None, None, None], [None, None, None], [None,
None, None]]
    for x in range(3):
        for y in range(3):
            button = PushButton(board, text="", grid=[x, y],
width=3, command=choose_square, args=[x,y])
            new_board[x][y] = button
    return new_board

def choose_square(x, y):
    board_squares[x][y].text = turn
    board_squares[x][y].disable()
    toggle_player()

def toggle_player():
    global turn
    if turn == [illegible]:
        turn = [illegible]
    else:
        turn = "X"
    message.value = "It is your turn, " + turn

# Variables -------------
turn = "X"

# App -------------------
app = App("Tic tac toe")

board = Box(app, layout="grid")
board_squares = clear_board()
message = Text(app, text="It is your turn, " + turn)

app.display()
```

Then add a new Text widget in the app section to display the turn.

```
message = Text(app, text="It is your turn,
" + turn)
```

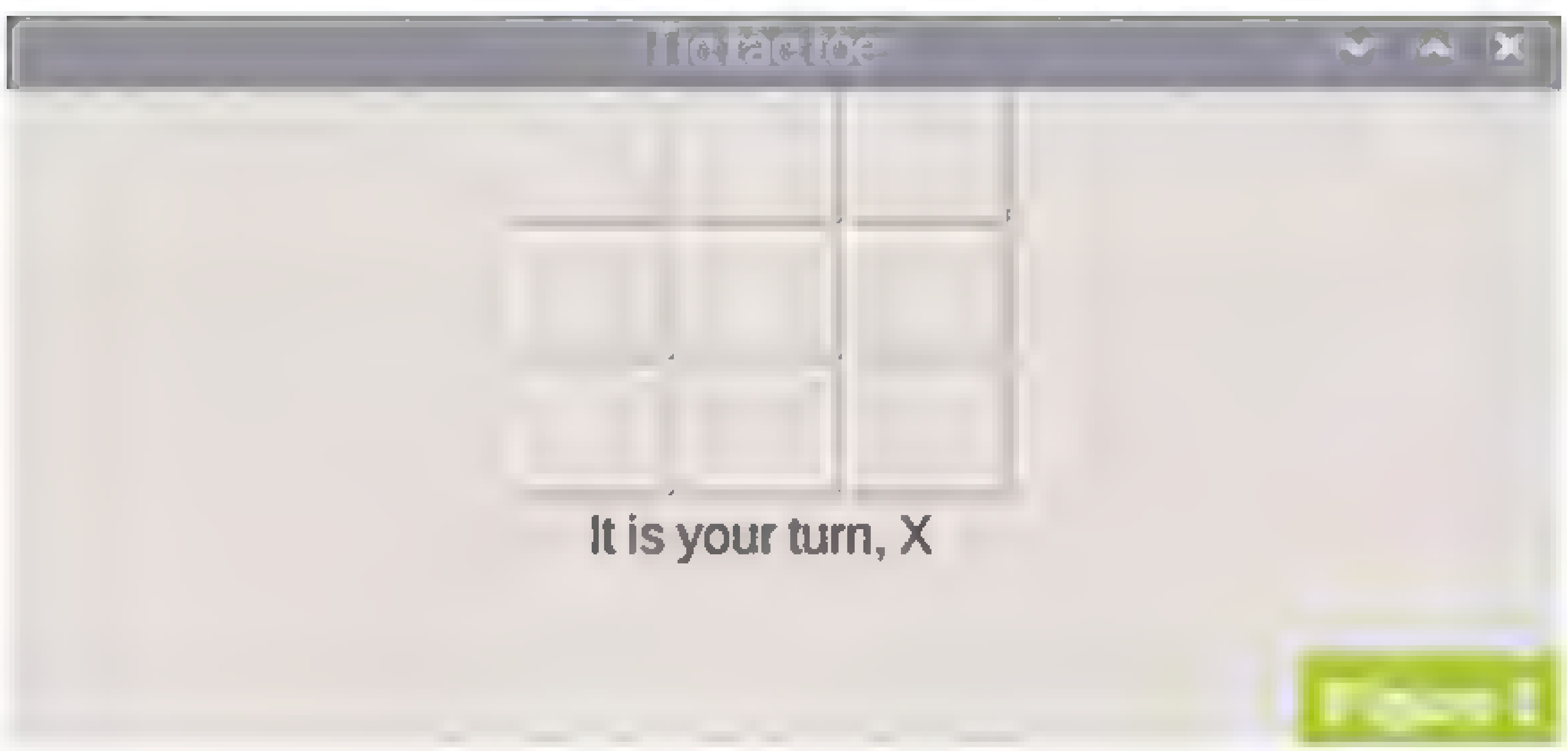


Figure 3 Let your players know whose turn it is

Move to the functions section and create a new function called `choose_square`.

```
def choose_square(x, y):
```

You will notice that this function takes two arguments – x and y. This is so that you know which square on the board has been clicked.

Add the following code (indented) inside the function to set the text inside the button that was clicked to the symbol of the current player, and then disable the button so it cannot be clicked on again.

```
board_squares[x][y].text = turn
board_squares[x][y].disable()
```

Finally, connect this function to the button. Find this line of code inside your `clear_board` function:

```
button = PushButton(board, text="",
grid=[x, y], width=3)
```

Modify it so that it looks like the line below:

```
button = PushButton(board, text="",
grid=[x, y], width=3, command=choose_square,
args=[x,y])
```

You have added two things here. Firstly, you are attaching a command, just as before. When the button is pressed, the function with this name will be called. Secondly, you are also providing arguments to this function, which are the co-ordinates x and y of the button which was pressed, so that you can find that button again in the list.

Your program should now look like **tictactoe3.py**. Save and run it. You will now be able to click on a button and it will change to an X. Unfortunately, in this version of the game it is permanently X's turn!

Alternating between players

Once one player has taken their turn, the `turn` variable should toggle to be the other player. Here is a function which will toggle from X to O and vice versa.

```
def toggle_player():
    global turn
    if turn == "X":
        turn = "O"
    else:
        turn = "X"
```

tictactoe5.py

> Language: **Python 3**

```
# Imports ---------------
from guizero import App, Box, PushButton, Text

# Functions -------------
def clear_board():
    new_board = [[None, None, None], [None, None, None], [None, None, None]]
    for x in range(3):
        for y in range(3):
            button = PushButton(board, text="", grid=[x, y], width=3, command=choose_square, args=[x,y])
            new_board[x][y] = button
    return new_board

def choose_square(x, y):
    board_squares[x][y].text = turn
    board_squares[x][y].disable()
    toggle_player()
    check_win()

def toggle_player():
    global turn
    if turn == "X":
        turn = "O"
    else:
        turn = "X"
    message.value = "It is your turn, " + turn

def check_win():
    winner = None

    # Vertical lines
    if (
        board_squares[0][0].text == board_squares[0][1].text == board_squares[0][2].text
    ) and board_squares[0][2].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[1][0].text == board_squares[1][1].text == board_squares[1][2].text
    ) and board_squares[1][2].text in ["X", "O"]:
        winner = board_squares[1][0]
    elif (
        board_squares[2][0].text == board_squares[2][1].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[2][0]

    # Horizontal lines
    elif (
        board_squares[0][0].text == board_squares[1][0].text == board_squares[2][0].text
    ) and board_squares[2][0].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[0][1].text == board_squares[1][1].text == board_squares[2][1].text
    ) and board_squares[2][1].text in ["X", "O"]:
        winner = board_squares[0][1]
    elif (
        board_squares[0][2].text == board_squares[1][2].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[0][2]

    # Diagonals
    elif (
        board_squares[0][0].text == board_squares[1][1].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[2][0].text == board_squares[1][1].text == board_squares[0][2].text
    ) and board_squares[0][2].text in ["X", "O"]:
        winner = board_squares[0][2]

    if winner is not None:
        message.value = winner.text + " wins!"

# Variables -------------
turn = "X"

# App -------------------
app = App("Tic tac toe")

board = Box(app, layout="grid")
board_squares = clear_board()
message = Text(app, text="It is your turn, " + turn)

app.display()
```

Add the code in your functions section. Notice the first line in the function: `global turn`. You need to specify this so that you are allowed to modify the global version of the `turn` variable, i.e. the one you already created. If you don't specify this, Python will create a local variable called `turn` and modify that instead, but that change won't be saved once the function exits.

You also need to make sure that the Text widget accurately reports the current player's turn. After the if/else statement in the `toggle_player` function, update the message like this:

```
message.value = "It is your turn, " + turn
```

Go back to your `choose_square` function and call the `toggle_player` function – with `toggle_player()` – once you have set the text and disabled the button. Your code should now resemble **tictactoe4.py**. Save and test the program again and you should find that you can click squares and they will alternately be designated either X or O.

Do we have a winner?

Finally, you need to write a function which will check whether there is a row, column, or diagonal of three Xs or Os, and if so will report the winner of the game.

Although it seems very inelegant, by far the easiest way to check if someone has won is to hard-code the checks for each vertical, horizontal, and diagonal line individually.

The following code is for one vertical line, one horizontal line, and one diagonal – can you add the rest?

```
def check_win():
    winner = None

    # Vertical lines
    if (
        board_squares[0][0].text == board_
squares[0][1].text == board_squares[0][2].text
    ) and board_squares[0][2].text in ["X", "O"]:
        winner = board_squares[0][0]

    # Horizontal lines
    elif (
        board_squares[0][0].text == board_
squares[1][0].text == board_squares[2][0].text
    ) and board_squares[2][0].text in ["X", "O"]:
        winner = board_squares[0][0]

    # Diagonals
    elif (
        board_squares[0][0].text == board_
squares[1][1].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[0][0]
```

Notice that the function begins by creating a Boolean variable called `winner`. If by the time the long if/elif statement has been executed, the value of this variable is True, you know that someone has won the game.

After adding the remaining winning line checks, add some code at the end of the function to change the display message if there has been a winner:

```
if winner is not None:
    message.value = winner.text + " wins!"
```

You now need to make sure that this function is called each time an X or O is placed, which corresponds to any time a button is pressed. Add a call to `check_win` at the end of the `choose_square` function, just in case the square that was chosen was the winning square.

Your program should now look like **tictactoe5.py**. Run it and test the game. If you wrote the tests in the `check_win` function correctly, you should find that the game detects correctly when a player has won.

Draw game

At the moment, the game will allow you to continue playing even after it has been won, until all of the squares are selected. It will also not tell you if the game was a draw. You could stop at this point, but if you really want to put the icing on the cake, adding a few more little touches could make your game more polished.

First, let's add some code to detect whether the game is a draw. The game is a draw if all of the squares contain either an X or an O, and no one has won. In the functions section, create a new function called `moves_taken`:

```
def moves_taken():
```

You're going to use this function to count the number of moves which have been made, so let's start a variable to keep count, initially beginning at 0.

```
def moves_taken():
  moves = 0
```

Now, remember when we created the `board_squares`, we used a nested loop to create all of the squares on the grid? We're going to need another nested loop here to check each and every square and determine whether it has been filled in with an X or O, or whether it is blank. Add this code for a nested loop to the `moves_taken` function:

```
for row in board_squares:
    for col in row:
```

Inside the loop, we need to check whether that particular square is filled in with an X or an O. If it is, add 1 to the `moves` variable to record that square has been counted

Global variables

It is arguably a bad idea to use global variables because if you have many functions in a large program, it can become extremely confusing as to which code modifies the value of a variable and when. In a small program like this, it is not too difficult to keep track.

Remember that it is possible to read and use the value of a global variable from inside a function without declaring it global, but in order to modify its value you will need to explicitly declare this. The functions in this program (and most GUI programs in this tutorial series) are actually modifying the values of your widgets as global variables. For example, when someone wins the game, you set the value of the message to display who won:

```
message.value = winner.text + " wins!"
```

In this example, `message` is a global variable. So how can we modify its value without declaring it as global? The answer is because we are using a *property* of the message widget, the property called value. Essentially what this code is saying is "Hey Python, you know that widget over there called message? Well, could you modify its value property please?" Python will allow modification through object properties in the global scope, but it won't allow you to directly modify the value of a variable without declaring it global.

Create Graphical User Interfaces with Python

For further tutorials on how to make your own GUIs with guizero, take a look at our book, *Create Graphical User Interfaces with Python*. Its 156 pages are packed with essential info and a range of exciting projects. **magpi.cc/pythongui**

06-tictactoe.py

> Language: **Python 3**

```python
# Imports
from guizero import App, Box, PushButton, Text

# Functions -------------
def clear_board():
    new_board = [[None, None, None], [None, None, None], [None, None, None]]
    for x in range(3):
        for y in range(3):
            button = PushButton(board, text="", grid=[x, y], width=3, command=choose_square, args=[x,y])
            new_board[x][y] = button
    return new_board

def choose_square(x, y):
    board_squares[x][y].text = turn
    board_squares[x][y].disable()
    toggle_player()
    check_win()

def toggle_player():
    global turn
    if turn ==     :
        turn =
    else:
        turn = "
    message.value = "It is your turn, " + turn

def check_win():
    winner = None

    # Vertical lines
    if (
        board_squares[0][0].text == board_squares[0][1].text == board_squares[0][2].text
    ) and board_squares[0][2].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[1][0].text == board_squares[1][1].text == board_squares[1][2].text
    ) and board_squares[1][2].text in ["X", "O"]:
        winner = board_squares[1][0]
    elif (
        board_squares[2][0].text == board_squares[2][1].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[2][0]

    # Horizontal lines
    elif (
        board_squares[0][0].text == board_squares[1][0].text == board_squares[2][0].text
    ) and board_squares[2][0].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[0][1].text == board_squares[1][1].text == board_squares[2][1].text
    ) and board_squares[2][1].text in ["X", "O"]:
        winner = board_squares[0][1]
    elif (
        board_squares[0][2].text == board_squares[1][2].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[0][2]

    # Diagonals
    elif (
        board_squares[0][0].text == board_squares[1][1].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[2][0].text == board_squares[1][1].text == board_squares[0][2].text
    ) and board_squares[0][2].text in ["X", "O"]:
        winner = board_squares[0][2]

    if winner is not None:
        message.value = winner.text + " wins!"
    elif moves_taken() == 9:
        message.value = "It's a draw"

def moves_taken():
    moves = 0
    for row in board_squares:
        for col in row:
            if col.text == "X" or col.text == "O":
                moves = moves + 1
    return moves


# Variables -------------
turn = "X"

# App -------------------
app = App("Tic tac toe")

board = Box(app, layout="grid")
board_squares = clear_board()
message = Text(app, text="It is your turn, " + turn)

app.display()
```

```python
if col.text == "X" or col.text == "O":
    moves = moves + 1
```

Finally, once the loops have finished, add a `return` statement to return the number of moves taken.

```python
return moves
```

Now, call this function inside the `check_win` function, to check for a draw. Add this code after the code that checks for a winner:

```python
if winner is not None:
    message.value = winner.text + " wins!"

# Add this code
elif moves_taken() == 9:
    message.value = "It's a draw"
```

Your code should resemble **06-tictactoe.py**. When run, the game will now check whether nine moves have been taken; if they have, it will change the message to report that the game was a draw.

How not to code: a guide to concise programming

Updating a 22-year-old game brought Andrew face to face with some very poor coding practices

AUTHOR
ANDREW GILLETT

Andrew Gillett is a tutor and programmer who has worked on ten published games including *RollerCoaster Tycoon 3*, *LostWinds 2*, and *Kinectimals*

Download the code from GitHub
wfmag.cc/wfmag48

In 1998, at the age of 17, I was learning how to write games in C. My first attempt, the subtly titled *DEATH*, was not going well. The game was my take on *Hardcore*, a 1992 Atari ST game by legendary developer and sheep enthusiast Jeff Minter, which had been released only as an unfinished five-level demo. The player controlled four gun turrets on the outside of a square arena, into which enemies teleported. While the original game had been enjoyable and promising, my version wasn't much fun, and I couldn't work out why. Making a decent game would also have involved making dozens of levels and many enemy types, which was looking like too big a task, especially as I was finding it hard to understand the intricacies of how the enemies in *Hardcore* moved.

So I abandoned that game and decided to replicate a different one – 1994's *MasterBlaster*, a *Bomberman*-style game on the Commodore Amiga. *MasterBlaster* didn't have a single-player mode or bots, so there was no enemy AI to write. And the level was just a grid with randomly generated walls and power-ups – so there was no real level design involved. With those two hurdles removed, development went fairly smoothly, the biggest challenge being working out some of the subtleties of how character movement worked.

The game, which I named *Partition Sector*, was finished in mid-1999 and spent the next 18 years on my website being downloaded by very few people. In late 2018 I decided to do a quick update to the game and release it on Steam. Then I started having ideas, and ended up working on it, on and off, for two years.

One of the biggest hurdles I came across when writing my first games was how to structure the code. I knew how to write a basic game loop, in which you update the positions of objects within the game, then draw the level and the objects within it, and then loop back to

A series of ultrabombs blowing up a snake.

The 2021 version of *Partition Sector*.

the start, ending the loop when the 'game over' criteria are met or the player has chosen to quit. But for a full game you need things like a main menu, submenus, going through a particular number of rounds before returning to the main menu, and so on. In the end, I was able to come up with something that worked, but looking back on my old code 20 years on, I could see many cases of absolutely terrible practice. While most of my time was spent adding new features, a lot of time was spent rewriting and restructuring old code. I'm going to share some examples from the original code so you don't make the same mistakes!

"I started having ideas, and ended up working on it, on and off, for two years"

AVOID REPETITION

In the original *Partition Sector* code, when the player moved the analogue stick up to move up the screen, there was a section of code which checked for this and then applied the movement. This included checks such as: is the player currently piloting a remote-control bomb, and is there space for the bomb to move up? If so, apply the movement. If the player isn't remote-controlling anything, check to see if there's space to move the player character up. This includes checks for either a wall or a bomb being in the way. If the player has the ghost power-up, they can walk through most, but not all, walls, so we have to check for that too. If the player's able to move, we move them up a small amount, then also do a small movement left or right if they aren't fully aligned with the vertical lane they're moving in. If movement is blocked by a wall or a bomb, and the player has the 'Strength' power-up which allows them to push such things, we apply the push effect, first making sure there's a free space for the wall or bomb to be pushed into. Finally, if the player is blocked by a wall, we might apply what I call secondary movement to the left or right, moving the player in the direction of an available lane. There's also code for updating the walking animation. In total, the code for the player moving up was 155 lines.

MOVEMENT LOGIC

Having written this code, I then proceeded to copy and paste it three times, for the down, left, and right directions. This is extremely bad. The code for moving down was identical to the code for moving up, except that it was checking for blockages and applying movement in the opposite direction. The same was true of the →

Wireframe

This tutorial first appeared in Wireframe, our sister magazine that lifts the lid on the world of video games. Every issue includes tutorials and in-depth interviews, along with news and reviews of the latest indie and triple-A games.

To find out more, visit their website at **wfmag.cc**.

Check out their subscription offers at **wfmag.cc/subscribe**.

OTHER LANGUAGES

Each programming language has its own way of giving you access to other code files – in C# and Java you can automatically use public classes from other files within the same project, while in C++ you have to use header files (with include guards or #pragma once) to declare shared functions and classes. When working in Python, note that it won't let you have circular imports – for example, writing `import file2` in **file1.py**, and `import file1` in **file2.py**.

left/right code, except it checked and updated the X axis instead. One of the biggest issues with this code was that any change to the logic had to be applied four times – once for each direction. Besides the increased workload, every time I made a change, there was a risk that I'd forget to apply it to one of the directions, in which case I'd end up with a character who obeyed different movement rules depending on which direction they were moving.

The solution was to create a function, `move`, with two integer parameters – `xDir` and `yDir`. A value of zero for either of these means 'Don't try to move on this axis', while values of -1 or 1 correspond to moving up/left or down/right. Each frame, I get the control inputs and call the function with the appropriate values. Inside `move`, instead of specifically checking for a wall above, below, or to the left or right of the player, I instead work out which grid square the player would be in if they were to move in the specified direction, then check to see if there's a wall or bomb in that square. Writing generic code like this can sometimes be slightly harder than writing separate code for each specific case, but it's a vital skill to practise. Part of what separates experienced and inexperienced programmers is the ability to recognise opportunities for reducing code duplication.

"Experienced programmers recognise opportunities for reducing code duplication"

The code in **Figure 1** and **Figure 2** shows two simplified recreations of the player movement code in Python. The second example has been split into multiple code files. In each case, grid squares are 64×64 pixels, and the player moves at a speed of two pixels per frame. In the first (bad!) code example, the update method in the `Player` class has separate movement code for each direction. If we take the example of trying to move left, we only allow the player to move if there isn't a wall immediately to their left. We just check a single point, which is halfway down the sprite on the Y axis, and just past its left edge on the X axis (in this example, the player's X and Y co-ordinates represent the top left of the sprite). We convert the selected pixel co-ordinates to grid co-ordinates and check for a wall – in the `GRID` list at the top of the code, each row is represented by a string, where a space represents an empty square, while 'X' represents a wall. If the proposed new position is empty, we apply the movement. We then check to see whether the player is perfectly aligned with their current horizontal lane. If not, we apply a small change to the Y position, moving it in the direction of being centred in the lane.

In **player.py** in the second set of example code, there's only one set of movement code, which deals with all four directions. You'll also notice that constants are used for the grid square size, half the grid square size, and the player movement speed. First, we set the variables `x_dir` and `y_dir` to -1, 0, or 1 to indicate which direction (if any) the player is trying to move on each axis. It uses the `Controls` classes, which are explained later on. The game doesn't allow for diagonal movement, so if the player's →

› Snake Hunter mode.

FIGURE 1

```
import pgzero, pgzrun

WIDTH,HEIGHT = 576,320

GRID = ['XXXXXXXXX',
        'X       X',
        'X X X X X',
        'X       X',
        'XXXXXXXXX']

class Player(Actor):
    def __init__(self,pos):
        super().__init__('player', pos,
anchor=('left', 'top'))

    def update(self):
        if keyboard.left:
            x = int(self.x) - 2
            y = int(self.y) + 64 // 2
            grid_x = x // 64
            grid_y = y // 64

            # The square ahead does not have a wall
            if GRID[grid_y][grid_x] == ' ':
                # Apply movement
                self.x -= 2

                # Lane alignment
                if y % 64 < 64 // 2:
                    self.y += 1
                elif y % 64 > 64 // 2:
                    self.y -= 1

        if keyboard.right:
            x = int(self.x) + 64 + 2 - 1
            y = int(self.y) + 64 // 2
            grid_x = x // 64
            grid_y = y // 64
            if GRID[grid_y][grid_x] == ' ':
                self.x += 2
                if y % 64 < 64 // 2:
                    self.y += 1
                elif y % 64 > 64 // 2:
                    self.y -= 1

        if keyboard.up:
            x = int(self.x) + 64 // 2
            y = int(self.y) - 2
            grid_x = x // 64
            grid_y = y // 64
            if GRID[grid_y][grid_x] == ' ':
                self.y -= 2
                if x % 64 < 64 // 2:
                    self.x += 1
                elif x % 64 > 64 // 2:
                    self.x -= 1

        if keyboard.down:
            x = int(self.x) + 64 // 2
            y = int(self.y) + 64 + 2 - 1
            grid_x = x // 64
            grid_y = y // 64
            if GRID[grid_y][grid_x] == ' ':
                self.y += 2
                if x % 64 < 64 // 2:
                    self.x += 1
                elif x % 64 > 64 // 2:
                    self.x -= 1

player = Player( (64,64) )

def update():
    player.update()

def draw():
    screen.clear()

    for row_index in range(len(GRID)):
        for column_index in range(len(GRID[row_
index])):
            if GRID[row_index][column_index] != ' ':
                x, y = column_index * 64, row_index
* 64

                screen.blit( [illegible] ,(x,y))

    player.draw()

pgzrun.go()
```

FIGURE 2

```
import pgzrun, pygame
from shared import *
from player import Player
from controls import KeyboardControls,
JoystickControls

WIDTH,HEIGHT = 576,320

controls = JoystickControls(0) if pygame.joystick.
get_count() > 0 else KeyboardControls()
player = Player( (64,64), controls )

def update():
    player.update()

def draw():
    screen.clear()

    for row_index in range(len(GRID)):
        for column_index in range(len(GRID[row_
index])):
            if GRID[row_index][column_index] != ' ':
                x, y = column_index * GRID_SQ_SIZE,
row_index * GRID_SQ_SIZE
                screen.blit('gridblock',(x,y))

    player.draw()

pgzrun.go()
```

Figure 1: An example of bad code. Sure, it works, but look at all that duplication.

Figure 2: An example of good code. Look how much more compact it is than the previous sample.

DIFFERENT CLASS

An abstract class is one that cannot be instantiated – or in other words, you can't create an object based on it. Its purpose is to serve as a base class for other classes which implement the specified abstract methods.

pressing keys for both axes at once, we ignore the up and down keys. If either `x_dir` or `y_dir` is non-zero, the player is trying to move. The variable `horizontal` exists purely for readability, and is set to `True` or `False` depending on whether the player is trying to move on the X axis. We calculate the centre position of the player, then work out which grid position we're going to check for a wall, based on the axis and direction. The function `get_grid_pos` is used to convert from pixel to grid co ordinates. If no wall is present, we apply the movement by updating both the X and Y co-ordinates based on `x_dir/y_dir` multiplied by `SPEED`. Although we only want to move on one axis, the variable for the other axis will be zero, so no movement will take place, without the need for an `if` statement. Finally, for lane alignment, we work out the position we'd like the player to be aligned with, and the `move_towards` function, defined, above the `Player` class, ensures that they will move in the necessary direction, without overshooting the target position.

SPLIT CODE INTO FUNCTIONS

The game's original C code contained a function named `playTheGame`, which was over 1200 lines long. This function was called when the player selected Start Game from the main menu, and handled setting up and playing each round of the game. Within this function, the largest section of code was the main game loop, which included updating each player (primarily dealing with movement and the dropping of bombs), updating bombs and explosions, updating the walls which shrink in towards the centre of the level, and drawing everything to the screen.

```
updatePlayers();
updateUncarriedFlags();
updateKOTH();
updateBombs();
updateExplosions();
updateDeath();
updateShrinking();
updateBattleRoyaleTeams();
updateRespawningWalls();
updateSnakeHunterSpawning();
```

> Moving code into separate functions makes it easier to see the big picture.

> Don't forget that a finished game needs menus.

It also checked to see if the round needed to end (because a player had won, for example). Apart from the code which updates bombs and explosions, none of this had been separated into separate functions. In the new version, the code and data for the game logic have been moved into a class named `Game`, and that class contains methods such as `updatePlayers`, `updateBombs`, `updateShrinkingWalls`, and so on. The code for the main game loop largely consists of calls to these new methods, making it shorter and more readable.

SPLIT YOUR CODE

The code for the 1999 version of the game consisted almost entirely of a single C++ file which was over 5500 lines long. At the time, I didn't know how to split my code into multiple files. These days I try to keep code files as short as possible – preferably no more than 1000 lines, although this isn't always possible. Ideally, each file should implement one class or module.

The improved version of the Python code shows how to split your code into multiple files. The main file is **maze_good.py** which runs the game, and uses the `from/import` statements to load in other Python files. The `Player` class is contained in **player.py**, the keyboard and gamepad controls are contained in **controls.py**, and **shared.py** contains constants and functions which need to be accessible from multiple files.

AVOID GLOBAL VARIABLES

The original code featured many variables that needed to be accessed from multiple functions. For example, there was an array of players, mainly accessed from the main game function, but also used in functions for getting the sprite for the current animation frame, drawing the players on the screen, and several others. My solution at the time was to use global variables, which can be accessed from any function. Beginner programming courses advise you to minimise your use of global variables, as they

King of the Hill mode features a scoring zone with flashing disco tiles.

can make it difficult to keep track of where variables are changed. Another problem with this approach is that it can lead to old data persisting. When I created a new variable, I had to remember to reset it each time a new game is started, otherwise the value of that variable from the previous game would carry over to the new game. Usually this led to obvious bugs which were easily fixed, but occasionally it led to more subtle bugs which weren't noticed for some time. Encapsulating these variables inside the `Game` class solved this issue, as the game object is recreated at the start of each game, ensuring that all variables contained within it are automatically set to their default values.

BEHAVIOUR AND CLASSES

Each player needs one or more variables indicating which controls they are using. For example, one player may be using a particular controller; another player may be using the arrow keys; a third player might be using different keys. In the original code was a variable indicating if the player was using a keyboard or a controller, and a variable specifying their controller or keyset number. Each time I wanted to check whether a player was pressing their controls in a particular direction, I would first have to check which type of controls they were using, and then had to check the input for the relevant control type.

"The 1999 version of the game consisted almost entirely of a single C++ file"

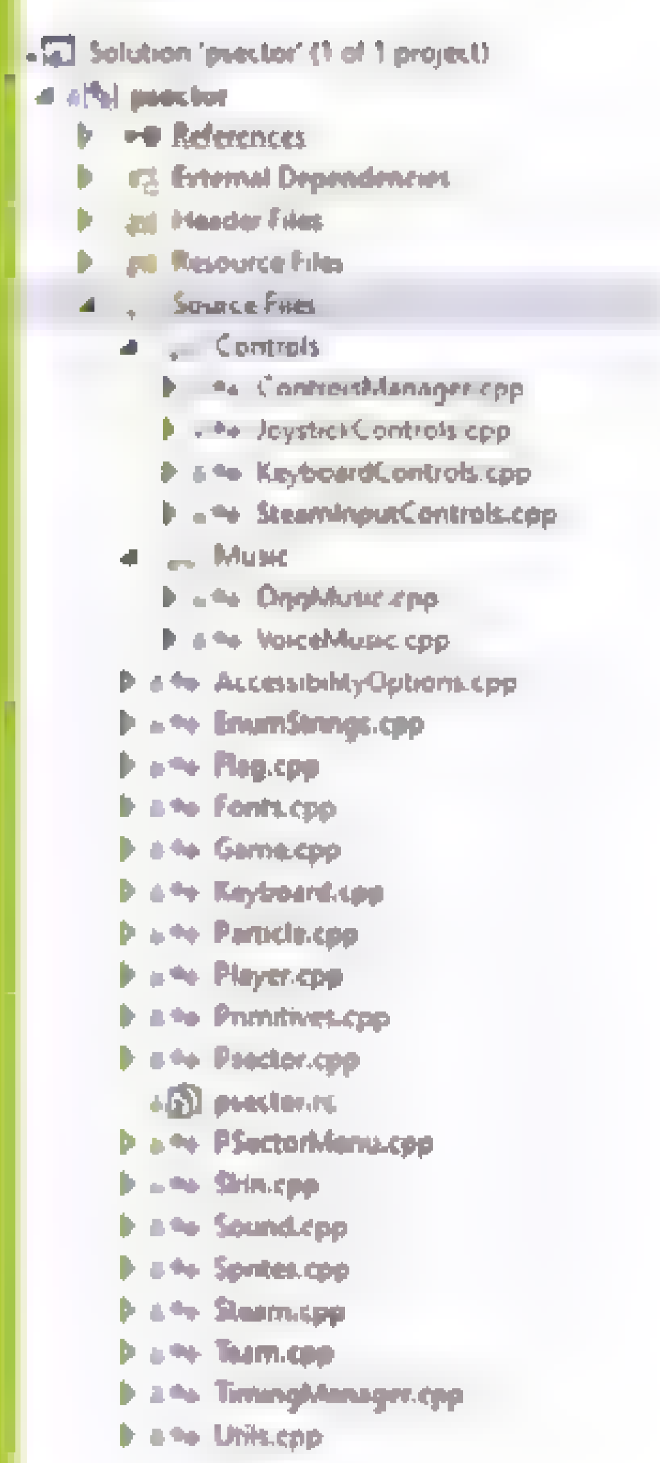
A better approach is to have a class named `Controls`, which exists purely as a base class for other classes that implement specific control methods. The `Controls` class itself isn't intended to be instantiated: it doesn't represent any particular control method, just controls in general. It specifies methods that its subclasses must override. In **controls.py**, the `Controls` class inherits from Python's `ABC` (abstract base class), and uses the `@abstractmethod` attribute to specify that any class inheriting from `Controls` must implement the methods `get_x_dir` and `get_y_dir`. The `KeyboardControls` and `JoystickControls` classes inherit from `Controls` and provide their own implementations of those methods. In the main file **maze_good.py**, we assign the player an instance of either `KeyboardControls` or `JoystickControls` depending on whether any controllers are connected.

The beauty of this system is the player code doesn't need to know anything about the different control methods, or even that a control method exists. All it cares about is having an object that it can call `get_x/y_dir` on. When I switched the game to using Steam's own input system for controllers, I created a new class called `SteamInputControls`. No changes needed to be made to the player code: I just had to give the players `SteamInputControls` objects instead of `JoystickControls` ones.

Writing bad code is inevitable when you're starting out – even experienced programmers sometimes do it. You don't have to write everything perfectly the first time, but if you learn from my mistakes and get into good habits early, you'll make your life a lot easier.

***Partition Sector* is out now on Steam.**

A list of C++ code files in the new version of *Partition Sector*.



Raspberry Pi Pico reaction game

Build a simple reaction timing game using an LED and push-buttons, for one or two players

Gareth Halfacree

With a passion for open-source software and hardware, Gareth was an early adopter of the Raspberry Pi platform and has written several publications on its capabilities and flexibility

@ghalfacree

Microcontrollers aren't only found in industrial devices: they power plenty of electronics around the home, including toys and games. In this tutorial you're going to build a simple reaction timing game, seeing who among your friends will be the first to press a button when a light goes off.

The study of reaction time is known as mental chronometry and while it forms a hard science, it is also the basis of plenty of skill-based games – including the one you're about to build. Your reaction time – the time it takes your brain to process the need to do something and send the signals to make that something happen – is measured in milliseconds: the average human reaction time is around 200–250 milliseconds, though some people enjoy considerably faster reaction times that will give them a real edge in the game!

For this project you'll need your Pico, a breadboard, an LED of any colour, a single 330 Ω resistor, two push-button switches, and a selection of male-to-male (M2M) jumper wires. You'll also need a micro USB cable, and to connect your Pico to your Raspberry Pi or other computer running the Thonny MicroPython IDE.

A single-player game

Start by placing your LED into your breadboard so that it straddles the centre divide. Remember that LEDs only work when they're the right way around: make sure you know which is the longer leg, or the anode, and which is the shorter leg, the cathode.

Using a 330 Ω current-limiting resistor, to protect both the LED and your Pico, wire the longer leg of the LED to pin GP15 at the bottom-left of your Pico as seen from the top with the micro USB cable uppermost. If you're using a numbered breadboard and have your Pico inserted at the very top, this will be breadboard row 20.

Take a jumper wire and connect the shorter leg of the LED to your breadboard's ground rail. Take another, and connect the ground rail to one of your Pico's ground (GND) pins – in **Figure 1**, we've used the ground pin on row three of the breadboard.

Next, add the push-button switch as shown in **Figure 1**. Take a jumper wire and connect one of the push-button's switches to pin GP14, right next to the pin you used for your LED. Use another jumper wire to connect the other leg – the one diagonally opposite the first, if you're using a four-leg push-button switch – to your breadboard's power rail. Finally, take a last jumper wire and connect the power rail to your Pico's 3V3 pin.

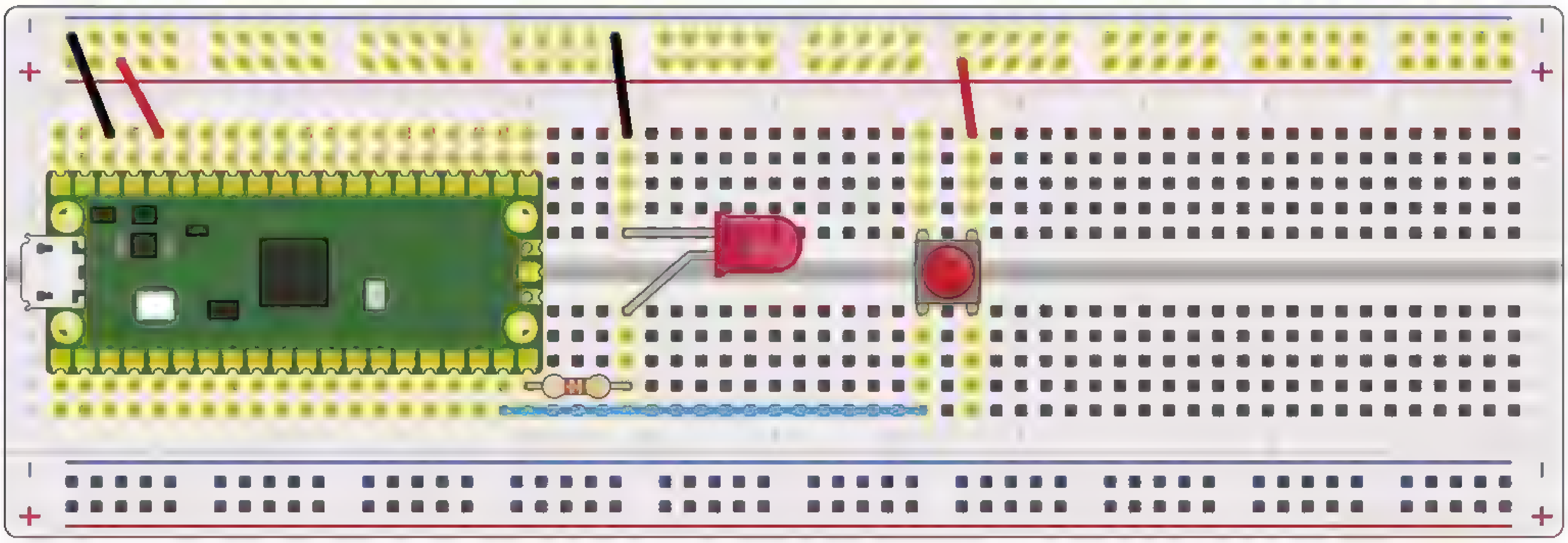
Your circuit now has everything it needs to act as a simple single-player game: the LED is the output device, taking the place of the TV you would normally use with a games console; the push-button switch is the controller; and your Pico is the games console, albeit one considerably smaller than you'd usually see!

Now you need to actually write the game. As always, connect your Pico to your Raspberry Pi or other computer and load Thonny. Create a new program, and start it by importing the machine library so you can control your Pico's GPIO pins:

```
import machine
```

You're also going to need the utime library:

```
import utime
```



In addition, you'll need a new library: urandom, which handles creating random numbers – a key part of making a game fun, and used in this game to prevent a player who has played it before from simply counting down a fixed number of seconds from clicking the Run button.

Next, set a pressed variable to False (more on this later) and set up the two pins you're using: GP15 for the LED, and GP14 for the push-button switch.

```
pressed = False
led = machine.Pin(15, machine.Pin.OUT)
button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)
```

In previous Pico tutorials, you've handled push-button switches in either the main program or in a separate thread. This time, though, you're going to take a different and more flexible approach: interrupt requests, or IRQs. The name sounds complex, but it's really simple: imagine you're reading a book, page by page, and someone comes up to you and asks you a question. That person is performing an interrupt request: asking you to stop what you're doing, answer their question, then letting you go back to reading your book.

A MicroPython interrupt request works in exactly the same way: it allows something, in this case the press of a push-button switch, to interrupt the main program. In some ways it's similar to a thread, in that there's a chunk of code which sits outside the main program. Unlike a thread, though, the code isn't constantly running: it only runs when the interrupt is triggered.

Start by defining a handler for the interrupt. This, known as a callback function, is the code which runs when the interrupt is triggered. As with any kind of nested code, the handler's code – everything after the first line – needs to be indented by four spaces for each level; Thonny will do this for you automatically.

```
def button_handler(pin):
    global pressed
    if not pressed:
        pressed=True
        print(pin)
```

This handler starts by checking the status of the **pressed** variable and then setting it to **True** to ignore further button presses (thus ending the game). It then prints out information about the pin responsible for triggering the interrupt. That's not too important at the moment – you only have one pin configured as an input, GP14, so the interrupt will always come from that pin – but lets you test your interrupt easily.

> You'll need a new library: urandom, which handles creating random numbers

Continue your program below, remembering to delete the indent that Thonny has automatically created – the following code is not part of the handler:

```
led.value(1)
utime.sleep(urandom.uniform(5, 10))
led.value(0)
```

Figure 1

▲ **Figure 1** A single-player reaction game

Warning!

Always remember that an LED needs a current-limiting resistor before it can be connected to your Pico. If you connect an LED without a current-limiting resistor in place, the best outcome is the LED will burn out and no longer work: the worst outcome is it could do the same to your Pico.

DOWNLOAD THE FULL CODE:

magpi.cc/github

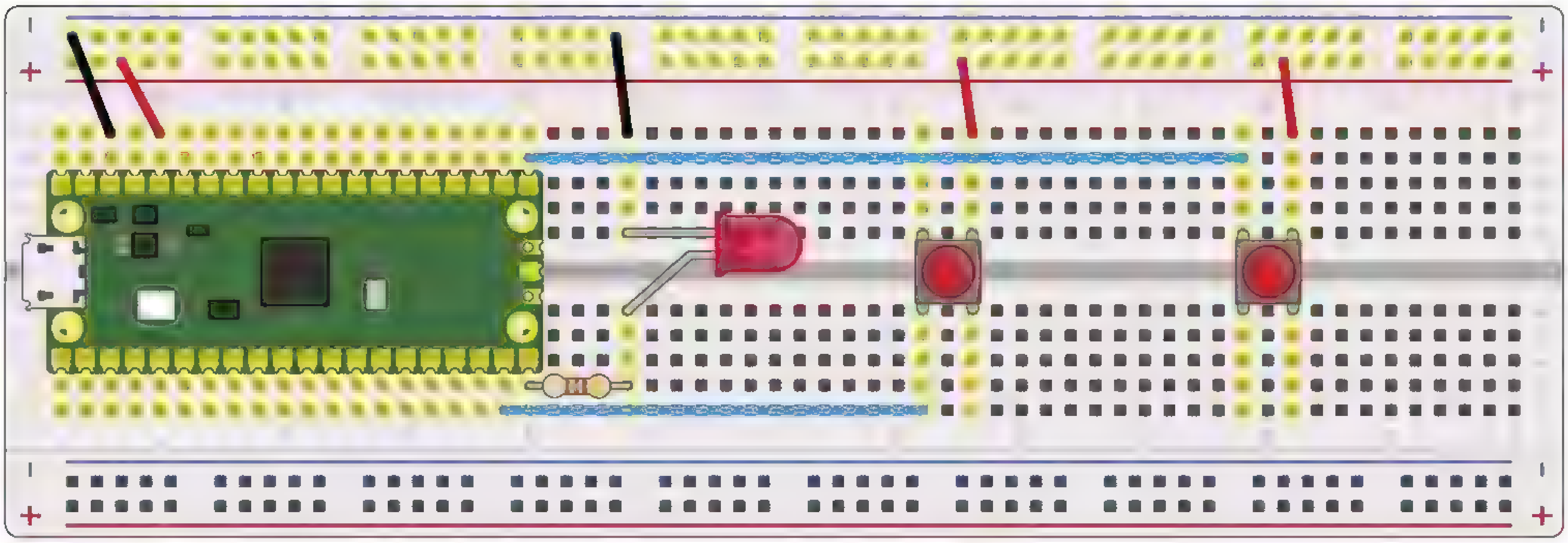


Figure 2

▲ **Figure 2** A two-player reaction game

This code will be immediately familiar to you: the first line turns the LED, connected to pin GP15, on; the next line pauses the program; the last line turns the LED off again – the player's signal to push the button. Rather than using a fixed delay, however, it makes use of the urandom library to pause the program for between five and ten seconds – the 'uniform' part referring to a uniform distribution between those two numbers.

At the moment, though, there's nothing watching for the button being pushed. You need to set up the interrupt for that, by typing in the following line at the bottom of your program:

```
button.irq(trigger=machine.Pin.IRQ_RISING,
handler=button_handler)
```

Setting up an interrupt requires two things: a trigger and a handler. The trigger tells your Pico what it should be looking for as a valid signal to interrupt what it's doing; the handler, which you defined earlier in your program, is the code which runs after the interrupt is triggered.

In this program your trigger is `IRQ_RISING`: this means the interrupt is triggered when the pin's value rises from low, its default state thanks to the built-in pull-down resistor, to high, when the button connected to 3V3 is pushed. A trigger of `IRQ_FALLING` would do the opposite: trigger the interrupt when the pin goes from high to low. In the case of your circuit, `IRQ_RISING` will trigger as soon as the button is pushed; `IRQ_FALLING` would trigger only when the button is released.

Your program should look like this:

```
import machine
import utime
import urandom

pressed = False
led = machine.Pin(15, machine.Pin.OUT)
button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)

def button_handler(pin):
    global pressed
    if not pressed:
        pressed=True
        print(pin)

led.value(1)
utime.sleep(urandom.uniform(5, 10))
led.value(0)
button.irq(trigger=machine.Pin.IRQ_RISING,
handler=button_handler)
```

Click the Run button and save the program to your Pico as **Reaction_Game.py**. You'll see the LED light up: that's your signal to get ready with your finger on the button. When the LED goes out, press the button as quickly as you can.

When you press the button, it triggers the handler code you wrote earlier. Look at the Shell area: you'll see your Pico has printed a message, confirming that the interrupt was triggered by pin GP14. You'll also see another detail: `mode=IN` tells you the pin was configured as an input.

That message doesn't make for much of a game, though: for that, you need a way to time the player's reaction speed. Start by deleting the line `print(pin)` from your button handler – you don't need it any more.

Go to the bottom of your program and add a new line, just above where you set up the interrupt:

```
timer_start = utime.ticks_ms()
```

This creates a new variable called **timer_start** and fills it with the output of the **utime.ticks_ms()** function, which counts the number of milliseconds that have elapsed since the utime library began counting. This provides a reference point: the time just after the LED went out and just before the interrupt trigger became ready to read the button press.

Next, go back to your button handler and add the following two lines, remembering that they'll need to be indented by four spaces so MicroPython knows they form part of the nested code:

```
    timer_reaction = utime.ticks_diff(utime.
ticks_ms(), timer_start)
    print("Your reaction time was " +
str(timer_reaction) + " milliseconds!")
```

The first line creates another variable, this time for when the interrupt was actually triggered – in other words, when you pressed the button. Rather than simply taking a reading from **utime.ticks_ms()** as before, though, it uses **utime.ticks_diff()** – a function which provides the difference between when this line of code is triggered and the reference point held in the variable **timer_start**.

The second line prints the result, but uses concatenation to format it nicely. The first bit of text, or string, tells the user what the number that follows means; the **+** means that whatever comes next should be printed alongside that string. In this case, what comes next is the contents of the **timer_reaction** variable – the difference, in milliseconds, between when you took the reference point for the timer and when the button was pushed and the interrupt triggered.

> Setting up an interrupt requires two things: a trigger and a handler

Finally, the last line concatenates one more string so the user knows the number is measured in milliseconds, and not some other unit like seconds or microseconds. Pay attention to spacing: you'll see that there's a trailing space after 'was' and before the end quote of the first string, and a leading space after the open quote of the second string and before the word 'milliseconds'. Without those, the concatenated string will print something like 'Your reaction time was323milliseconds'.

Your program should now look like this:

```
import machine
import utime
import urandom
pressed = False
led = machine.Pin(15, machine.Pin.OUT)
button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)

def button_handler(pin):
    global pressed
    if not pressed:
        pressed=True
        timer_reaction = utime.ticks_
diff(utime.ticks_ms(), timer_start)
        print("Your reaction time was " +
str(timer_reaction) + " milliseconds!")

led.value(1)
utime.sleep(urandom.uniform(5, 10))
led.value(0)
timer_start = utime.ticks_ms()
button.irq(trigger=machine.Pin.IRQ_RISING,
handler=button_handler)
```

Click the Run button again, wait for the LED to go out, and push the button. This time, instead of a report on the pin which triggered the interrupt, you'll see a line telling you how quickly you pushed the button – a measurement of your reaction time. Click the Run button again and see if you can push the button more quickly this time – in this game, you're trying for as low a score as possible!

A two-player game

Single-player games are fun, but getting your friends involved is even better. You can start by inviting them to play your game and comparing your high – or, rather, low – scores to see who has the quickest reaction time. Then, you can modify your game to let you go head-to-head!

Start by adding a second button to your circuit. Wire it up the same as the first button, with one leg going to the power rail of your breadboard but with the other going to pin GP16 – the pin across the board from GP14 where the LED is connected, at the opposite corner of your Pico. Make sure the two buttons are spaced far enough apart that each player has room to put their finger on their button. Your finished circuit should look like **Figure 2**.

Although your second button is now connected to your Pico, it doesn't know what to do with it yet.

Go back to your program in Thonny and find where you set up the first button. Directly beneath this line, add:

```
right_button = machine.Pin(16, machine.Pin.
IN, machine.Pin.PULL_DOWN)
```

You'll notice that the name now specifies which button you're working with: the right-hand button on the breadboard. To avoid confusion, edit the line above so that you make it clear what was the only button on the board is now the left-hand button:

```
left_button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)
```

You'll need to make the same change elsewhere in your program, too. Scroll to the bottom of your code and change the line that sets up the interrupt trigger to:

```
left_button.irq(trigger=machine.Pin.IRQ_
RISING, handler=button_handler)
```

Add another line beneath it to set up an interrupt trigger on your new button as well:

```
right_button.irq(trigger=machine.Pin.IRQ_
RISING, handler=button_handler)
```

Your program should now look like this:

```
import machine
import utime
import urandom

pressed = False
led = machine.Pin(15, machine.Pin.OUT)
left_button = machine.Pin(14, machine.Pin.IN,
machine.Pin.PULL_DOWN)
right_button = machine.Pin(16, machine.Pin.IN,
machine.Pin.PULL_DOWN)

def button_handler(pin):
    global pressed
    if not pressed:
        pressed=True
        timer_reaction = utime.ticks_
diff(utime.ticks_ms(), timer_start)
        print("Your reaction time was " +
str(timer_reaction) + " milliseconds!")

led.value(1)
utime.sleep(urandom.uniform(5, 10))
led.value(0)
timer_start = utime.ticks_ms()
right_button.irq(trigger=machine.Pin.IRQ_RISING,
handler=button_handler)
left_button.irq(trigger=machine.Pin.IRQ_RISING,
handler=button_handler)
```

Click the Run icon, wait for the LED to go out, then press the left-hand push-button switch: you'll see that the game works the same as before, printing your reaction time to the Shell area. Click the Run icon again, but this time when the LED goes out, press the right-hand button: the game will work just the same, printing your reaction time as normal.

To make the game a little more exciting, you can have it report on which of the two players was the first to press the button. Go back to the top of your program, just below where you set up the LED and the two buttons, and add the following:

```
fastest_button = None
```

This sets up a new variable, **fastest_button**, and sets its initial value to **None** – because no button has yet been pressed. Next, go to the bottom of your button handler and delete the two lines which handle the timer and printing – then replace them with:

```
global fastest_button
    fastest_button = pin
```

Remember that these lines will need to be indented by four spaces so that MicroPython knows they're part of the function. These two lines allow your function to change, rather than just read, the **fastest_button** variable, and set it to contain the details of the pin which triggered the interrupt – the same details your game printed to the Shell area earlier in the tutorial, including the number of the triggering pin.

Now go right to the bottom of your program, and add these two new lines:

```
lhile fastest_button is None:
    utime.sleep(1)
```

This creates a loop, but it's not an infinite loop. here, you've told MicroPython to run the code in the loop only when the **fastest_button** variable is still zero – the value it was initialised with at the start of the program. In effect, this pauses

your program's main thread until the interrupt handler changes the value of the variable. If neither player presses a button, the program will simply pause.

Finally, you need a way to determine which player won – and to congratulate them. Type the following at the bottom of the program, making sure to delete the four-space indent Thonny will have created for you on the first line – these lines do not form part of the loop:

```
if fastest_button is left_button:
    print("Left Player wins!")
elif fastest_button is right_button:
    print("Right Player wins!")
```

The first line sets up an 'if' conditional which looks to see if the `fastest_button` variable is `left_button` – meaning the IRQ was triggered by the left-hand button. If so, it will print a message – with the line below indented by four spaces so that MicroPython knows it should run it only if the conditional is true – congratulating the left-hand player, whose button is connected to GP14.

The next line, which should not be indented, extends the conditional as an 'elif' – short for 'else if', a way of saying 'if the first conditional wasn't true, check this conditional next'. This time it looks to see if the `fastest_button` variable is `right_button` – and, if so, prints a message congratulating the right-hand player, whose button is connected to GP16.

Your finished program should look like this:

```
import machine
import utime
import urandom

pressed = False
led = machine.Pin(15, machine.Pin.OUT)
left_button = machine.Pin(14, machine.Pin.
IN, machine.Pin.PULL_DOWN)
right_button = machine.Pin(16, machine.Pin.
IN, machine.Pin.PULL_DOWN)
fastest_button = None

def button_handler(pin):
    global pressed
    if not pressed:
        pressed=True
        global fastest_button
        fastest_button = pin

led.value(1)
utime.sleep(urandom.uniform(5, 10))
led.value(0)
timer_start = utime.ticks_ms()
left_button.irq(trigger=machine.Pin.IRQ_
RISING, handler=button_handler)
right_button.irq(trigger=machine.Pin.IRQ_
RISING, handler=button_handler)

while fastest_button is None:
    utime.sleep(1)
if fastest_button is left_button:
    print("Left Player wins!")
elif fastest_button is right_button:
    print("Right Player wins!")
```

Press the Run button and wait for the LED to go out – but don't press either of the push-button switches just yet. You'll see that the Shell area remains blank, and doesn't bring back the >>> prompt; that's because the main thread is still running, sitting in the loop you created.

Now push the left-hand button, connected to pin GP14. You'll see a message congratulating you printed to the Shell – your left hand was the winner! Click Run again and try pushing the right-hand button after the LED goes out: you'll see another message printed, this time congratulating your right hand. Click Run again, this time with one finger on each button: push them both at the same time and see whether your right hand or left hand is faster!

Now that you've created a two-player game, you can invite your friends to play along and see which of you has the fastest reaction times!

Get Started with MicroPython on Raspberry Pi Pico

For more physical computing projects to try on your Raspberry Pi Pico, grab a copy of the new book, *Get Started with MicroPython on Raspberry Pi Pico*. As well as learning how to use Raspberry Pi Pico's pins as inputs and outputs, you'll build a simple game, measure temperatures, save and load data to your Pico's file system, and even make a burglar alarm for your room. *Get Started with MicroPython on Raspberry Pi Pico* is available now from **magpi.cc/picobook**

Cheap Trills for all

Add extra functionality and options to your Trill Guitar

MAKER

Mike Cook

Veteran magazine author from the old days, writer of the Body Build series, plus co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.

magpi.cc/mikecook

Having seen last month how to use a Raspberry Pi Pico controller, now we turn to what you need to do to make it play, not only like a proper guitar, but also one on which it's easy to play preprogrammed songs.

01 Guitar modes

Our MIDI Guitar uses three modes: configuration, free play, and song play. The configuration mode is entered by changing the configuration / play switch, and it allows us to change a few parameters and to select what 'set' the guitar is configured to use. The sets are defined by a configuration file in Raspberry Pi Pico's memory; this is easy to modify, as we shall see. Basically, the free play sets allow you to change the chords by touching a position, called a fret, on the bar sensor. Whereas the song play mode moves to the next chord with a single touch – see **Figures 1**, **2**, and **3**.

02 Trill library

While we introduced the Python Trill library in *The MagPi* #102 (**magpi.cc/102**), this will not run directly under MicroPython because of the different I2C library used by the latter. In addition, the Pico development notes say that addressing non-existent I2C hardware sometimes locks up your Pico. Finally, Pico allows six different pairs of pins for each of the two I2C buses, and it can easily change the I2C speed. Therefore, we've written a Pico version of the Trill library to allow you to select the pins to use, along with the I2C speed, when you initialise the sensors. It is called **pico_trill_lib.py**.

03 Installing the software

First, install the OLED display drivers. To do this, open up Thonny with the guitar connected and press the Stop icon to get the Python prompt. Then go to menu Tools > Manage plug-ins, type

Figure 1 Configuration display on the head of the guitar

Figure 2 Free play display with selected chord in a box

DOWNLOAD THE FULL CODE:

magpi.cc/pibakery

'ssd1360' into the search window, then click 'Search on PyPi'. It will return with a list of possible matches; click on the 'micropython-ssd1327' option and then click 'Install'. Open **pico_trill_lib.py** in Thonny, and choose 'Save as…' from the File menu, then click on the 'Raspberry Pi Pico' box to save it on Pico. Do the same with the **guitarConfigure.txt** file. Finally, load the **midi_trill_guitar.py** file and save this as **main.py**.

04 Playing the guitar

Connect your guitar to a MIDI sound generator, either directly or through a MIDI interface cable. Disconnect any USB connections

and slide the power switch to ON. You will see the first set come up 'Free play 1'; swipe across the square pad and you will hear the chord of A minor ring out. Note: if you swipe at the bottom, the sound will be quiet, but swiping across the top will produce full volume. See the effects of a diagonal swipe across the square pad. Tap your finger at any position to get just a single guitar string to sound.

05 Star-spangled fretboard

At this stage, we realised it would be a good idea to mark the points on the bar sensor that corresponded to each chord, to make it easy to pick out the point to press. So we slid our finger up and down the bar sensor so we got to a point that was equal distance between changing the chord up or down, and placed a sticker there. As there are some fret positions that have special uses in the song and configuration mode, we used stars for these positions. **Figure 4** (overleaf) shows how we positioned these stickers.

06 Changing sets

The example configuration file contains other sets. To change the set, move the slide

You'll Need

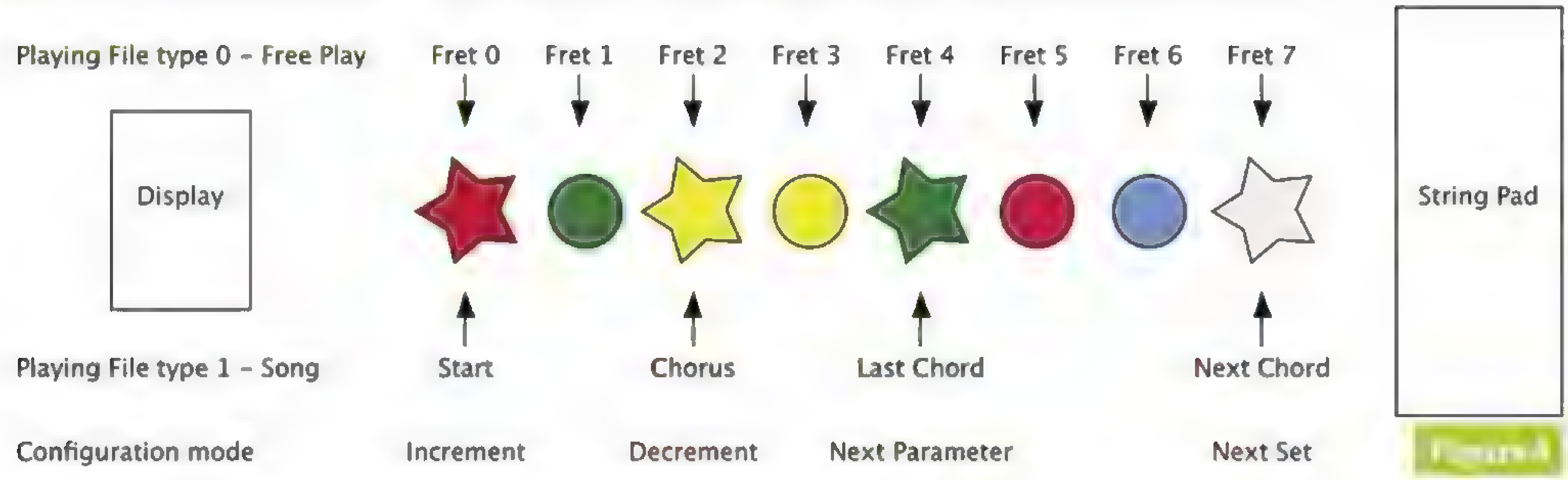
- Dot and star stickers **magpi.cc/stickers**
- MIDI Trill Guitar **magpi.cc/104**

Top Tip

Left-hand guitar

To make this a left-handed guitar, change the variable in line 31 of the code to True.

Figure 3 Song play display shows three lines and scrolls if there are more



▲ **Figure 4** Fret position meanings and usage

▼ **Figure 5** File type 0, free play configuration format

Figure 6 File type 1, song configuration format

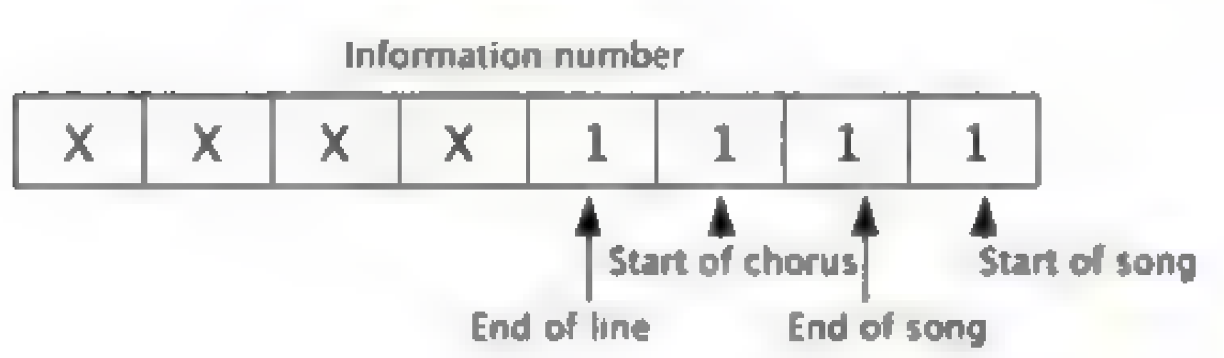
switch from play to configure and then tap the bottom fret, silver star, to step through the sets. Note that the set selected only moves in one direction and loops round to the start after the last set is displayed. Moving the switch back to play, the guitar will now have a new set of chords for free play, or chord sequence for a song set. The parameters for the chosen set are also applied to the guitar. You can create your own sets by editing the **guitarConfigure.txt** file.

File entry	Meaning
Free play 1, 12	# first line name max 16 charactors & relitave link to next set
0, 0	# second line file type & MIDI Channel
0, 800	# third line capo value and sustain note time before sending note off message
24, 0	# fourth line voice change, bank LSB
0, 11	#fret number 0 - chord number Am
1, 6	#fret number 1 - chord number C
2, 7	#fret number 2 - chord number D
3, 2	#fret number 3 - chord number F
4, 1	#fret number 4 - chord number E
5, 3	#fret number 5 - chord number G
6, 8	#fret number 6 - chord number Em
7, 4	#fret number 7 - chord number A

For file type 0
Free play 1

File entry	Meaning
House Rising Sun , 18	# first line name max 16 charactors & relitave link to next set
1, 0	# second line file type & MIDI Channel (0 = channel 1)
0, 800	# third line capo value and sustain note time before sending note off message
24, 0	# fourth line voice change, bank LSB
1, 11	#info start of song - chord number Am
0, 6	#information number - chord number C
0, 7	#information number - chord number D
8, 2	#info last in the line - chord number F
0, 11	#information number - chord number Am
0, 6	#information number - chord number C
8, 1	#info last in the line - chord number E
0, 11	#information number - chord number Am
0, 6	#information number - chord number C
0, 7	#information number - chord number D
8, 2	#info last in the line - chord number F
0, 11	#information number - chord number Am
0, 1	#information number - chord number E
10, 11	#info last in the line - chord number Am & end of song

Line 1
Line 2
Line 3
Line 4
For file type 1
Song file



07 Making your own sets

Figure 5 shows the format of a free play set. These sets are blocks of data in a format known as a linked list. At the start of each set is a link to the next set; we use the number of lines to the next set. As song sets can be any length, a linked list is essential. **Figure 6** shows a typical song set, where the first number on a line can have a special meaning and the second number is the chord number. An optional single chorus marker can be placed in a song file for those songs that end with several repeated choruses.

08 Configuration parameters

These are defined in the first four lines of a set entry, but you can temporarily override them by entering the configuration mode. The green star steps through all four options, while the red and yellow stars allow you to increment or decrement the values. A capo is a device that clamps over the neck of a guitar, holding all the strings down across a fret; this changes the key of the chords you play by, in effect, shortening all the strings. However, unlike a real guitar, here you can also have a negative value for the capo, which in effect lengthens the strings.

09 Sustain

The sustain parameter needs a bit of explanation. **Figure 7** shows the normal development of a sound between a note-on and note-off message. But in order to make the guitar feel like a guitar, and not a keyboard, the note-on message is sent when a touch is removed, not when it is detected. This makes an enormous difference to how you play it. So, in order to send a matching note-off message, the software automatically sends it a certain time after the touch is released. This time is set by the sustain value, which is in milliseconds.

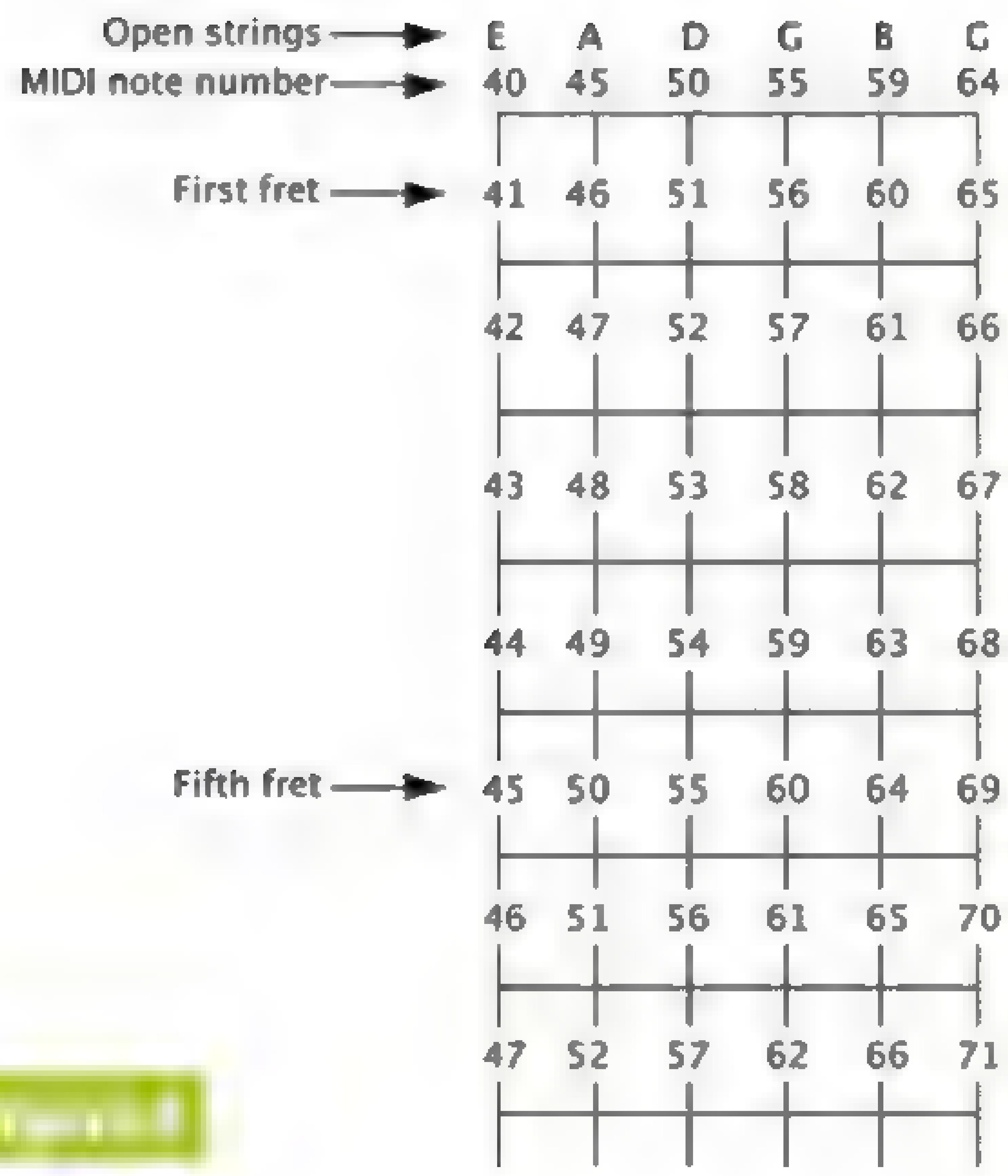


Figure 8

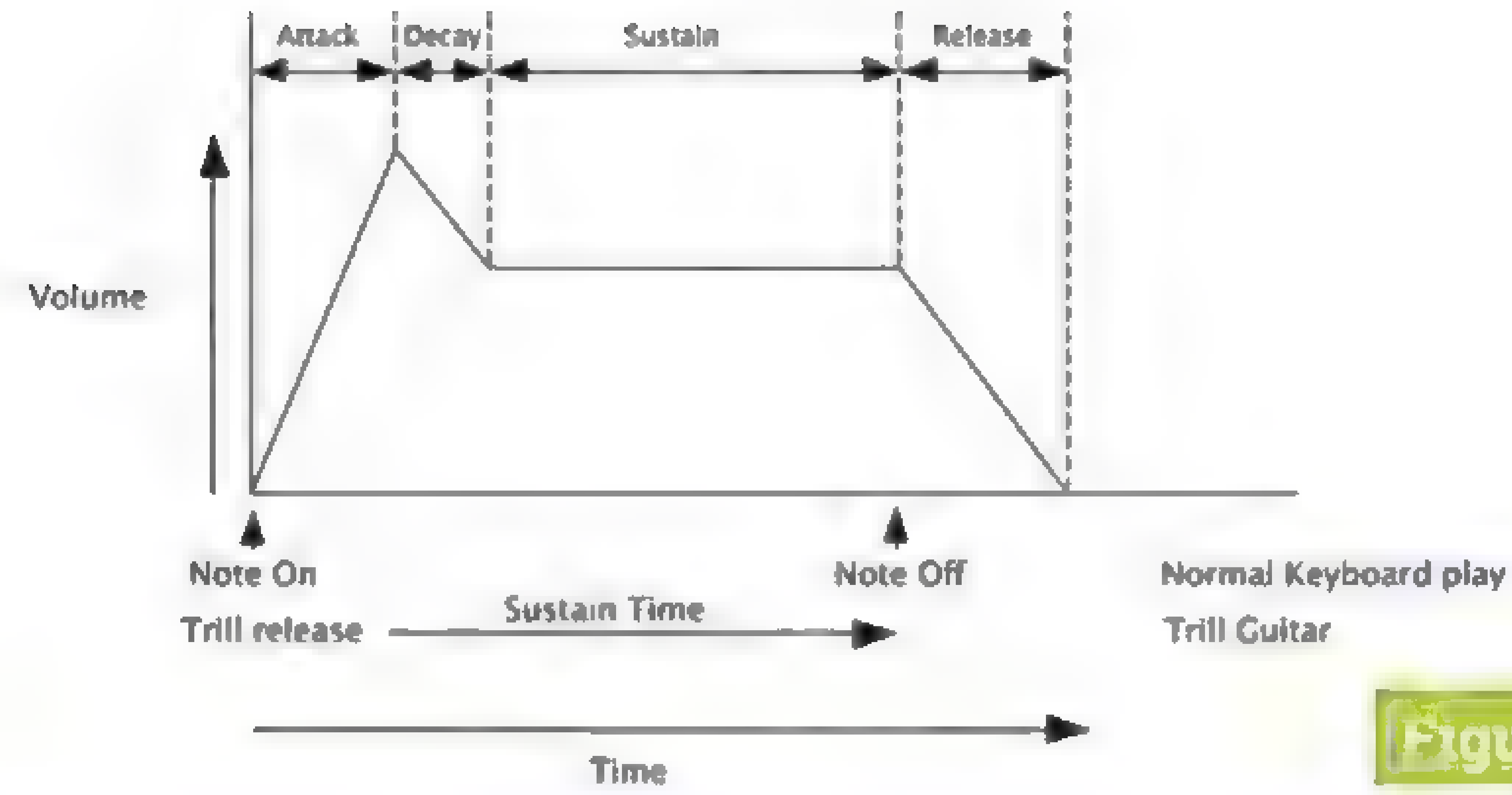


Figure 7

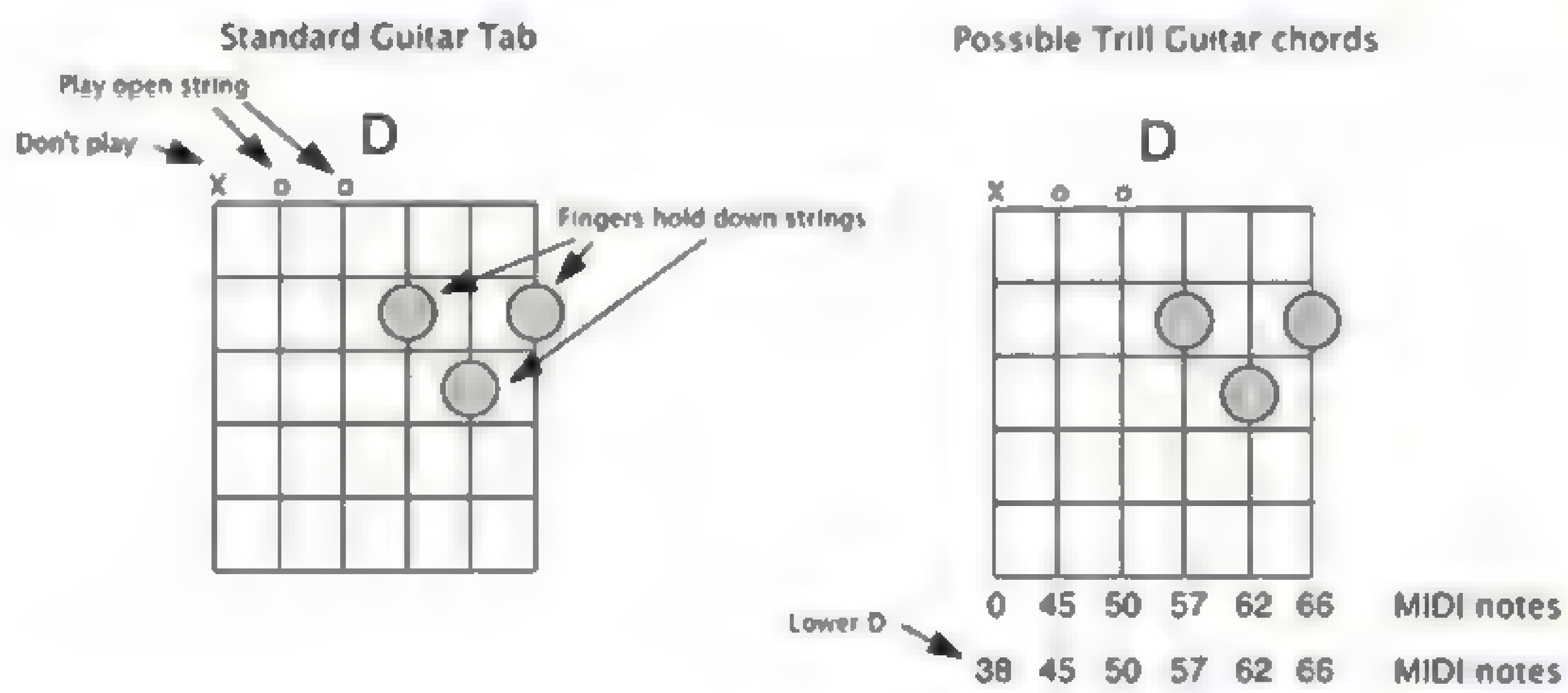


Figure 9

10 Chords

There are many hundreds of chords possible, but only 48 are defined in the software. You can add extra ones after line 92. The relationship between MIDI note numbers and frets on a real guitar is shown in **Figure 8**. Chord patterns are normally shown as a tab, and this needs to be translated into MIDI values; **Figure 9** shows how this is done. Note that some strings have an X above them, indicating that the string should not be plucked. If you place a zero here, there will be no note sounded, but we prefer to add an extra note, normally the next chord root down.

11 Songs

The internet is full of songs written in guitar tab format, many of them for a limited number of chords – just search for 'guitar strum songs for beginners'. A guitar tab is a list of lyrics with the chord changes above each specific word, or part way through the word. It is quite easy to generate a song set entry in your **guitarConfig.txt** file using these, and the chord numbers shown in lines 44 to 92. We would recommend you also generate and print out a text file showing the words, and chords, to use while you learn the song.

12 Strum or pick

With the song mode, all you have to do is to touch one star at the right time to automatically get the next chord. This leaves the other hand to strum in time, or pick individual notes by tapping your fingers. We found it sounds good whatever single strings you tap – just tap in the correct rhythm. You can also use a mixture of strumming and tapping. We have found this is a great way for absolute beginners on the guitar to get good results, and improvement can be rapid with a few minutes' practice.

Figure 7 Difference between playing a keyboard and playing a guitar

Figure 8 Guitar frets seen as MIDI note numbers

Figure 9 Converting a tab chord notation to MIDI note numbers

#MonthOfMaking 2021 SHOWCASE

What did readers of The MagPi get up to in March? By **Rob Zwetsloot**

Every March we put out a call – make something, and show us! There are very few limits and rules, so we end up getting some cool stuff sent to us.

We saw an amazing variety of stuff made by our community this year, so we thought it would be a great idea to show off what they'd made.

> We saw an amazing variety of stuff

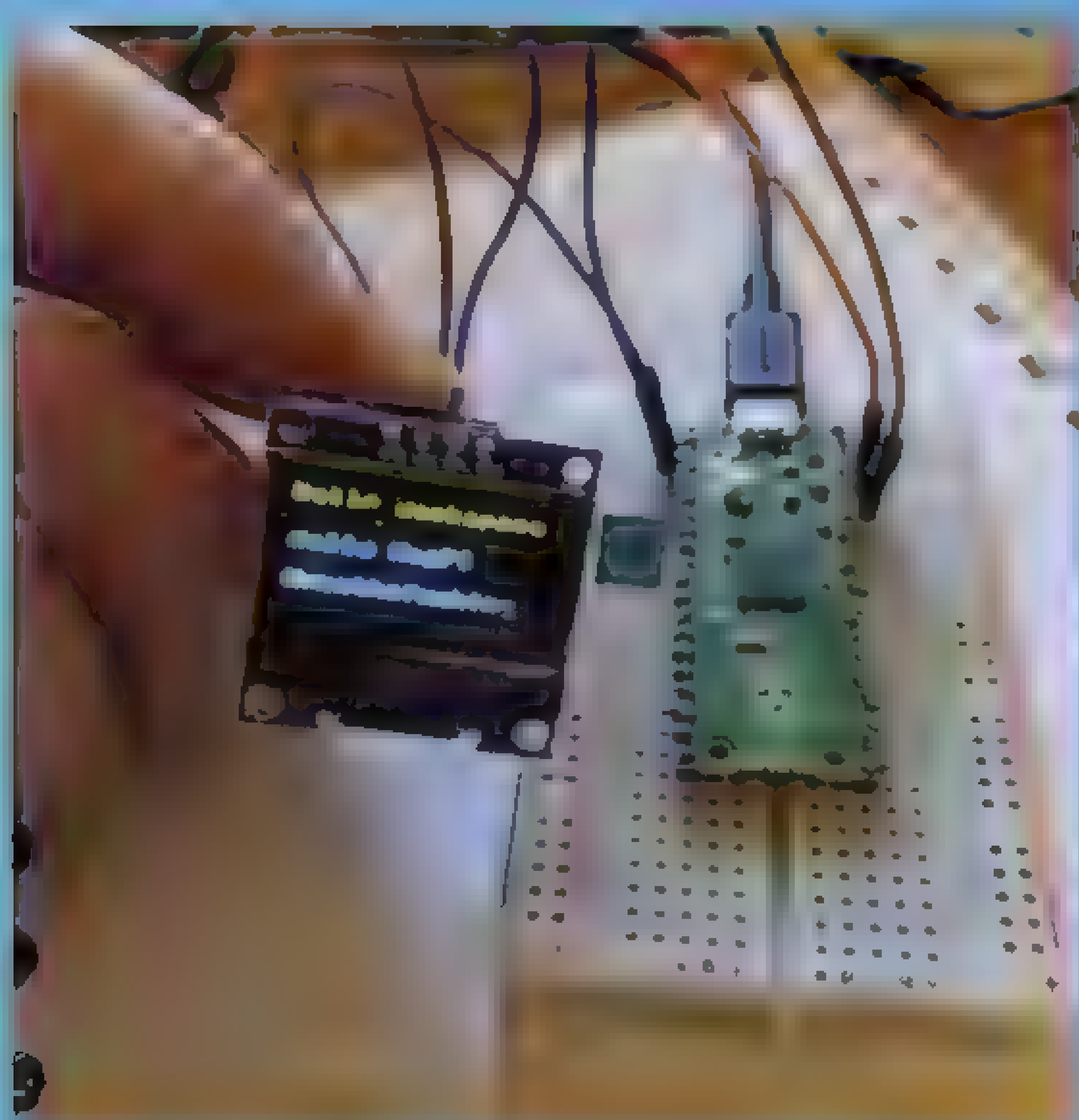
Remember, you can keep making and showing us what you've done any day of the year! We look forward to seeing what else the community makes over the rest of the year.

Pico MIDI Fighter

Here comes a new challenger

MIDI controllers are cool, programmable music boards. What makes this a MIDI Fighter? Well, the use of arcade buttons that you might find on a Street Fighter cabinet. They can take a hit, making for some excellent and frantic tunes.

Liz aka Blitz City DIY



Raspberry Pi Pico Lookup Display

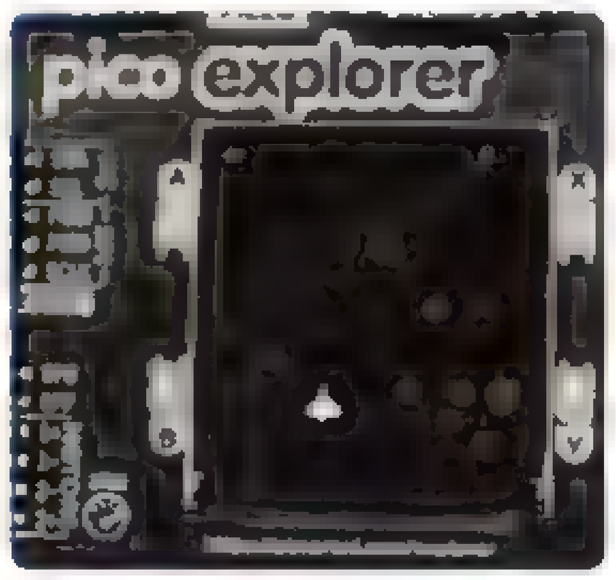
A tiny database

We were already pretty impressed by the use of the tiny SSD1306 display on Pico, but Edward went a step further and attached a tiny wireless chip as well so that it can connect to the internet. This could make for a cool watch-style project.

EdwardJ

More to come

Don't worry if you didn't finish your project in March, you can make any time of the year



Raspberry Pi Pico – space game

"As it's the last day for #MonthOfMaking, here is a preview of my Raspberry Pi Pico Space Game. Still working on the write-up. A graphical game for the Pico Explorer coded in MicroPython. Using both bitmap sprites and vector images."

Stewart Watkiss aka Penguin Tutor @penguintutor

Stewart finished the write up before we went to press: ***magpi.cc/picospace***

Designing projects

"I seem to have morphed from a #Maker into a #Designer. So most of my #MonthOfMaking has been spent in electronics & CAD design on the computer, reading datasheets, and sourcing components. I know this is all part of the making process, but it would be good to actually make something!"

Dr Footleg @drfootleg

We look forward to seeing what you make from these!

Fiona

New look voice assistant

This project looks very familiar – probably because it's a Google AIY Voice Kit, like the one we gave away with issue 57 all those years ago. This one has had a bit of a paint job, so it has its own unique look and personality. We hope Fiona helps Edgardo.

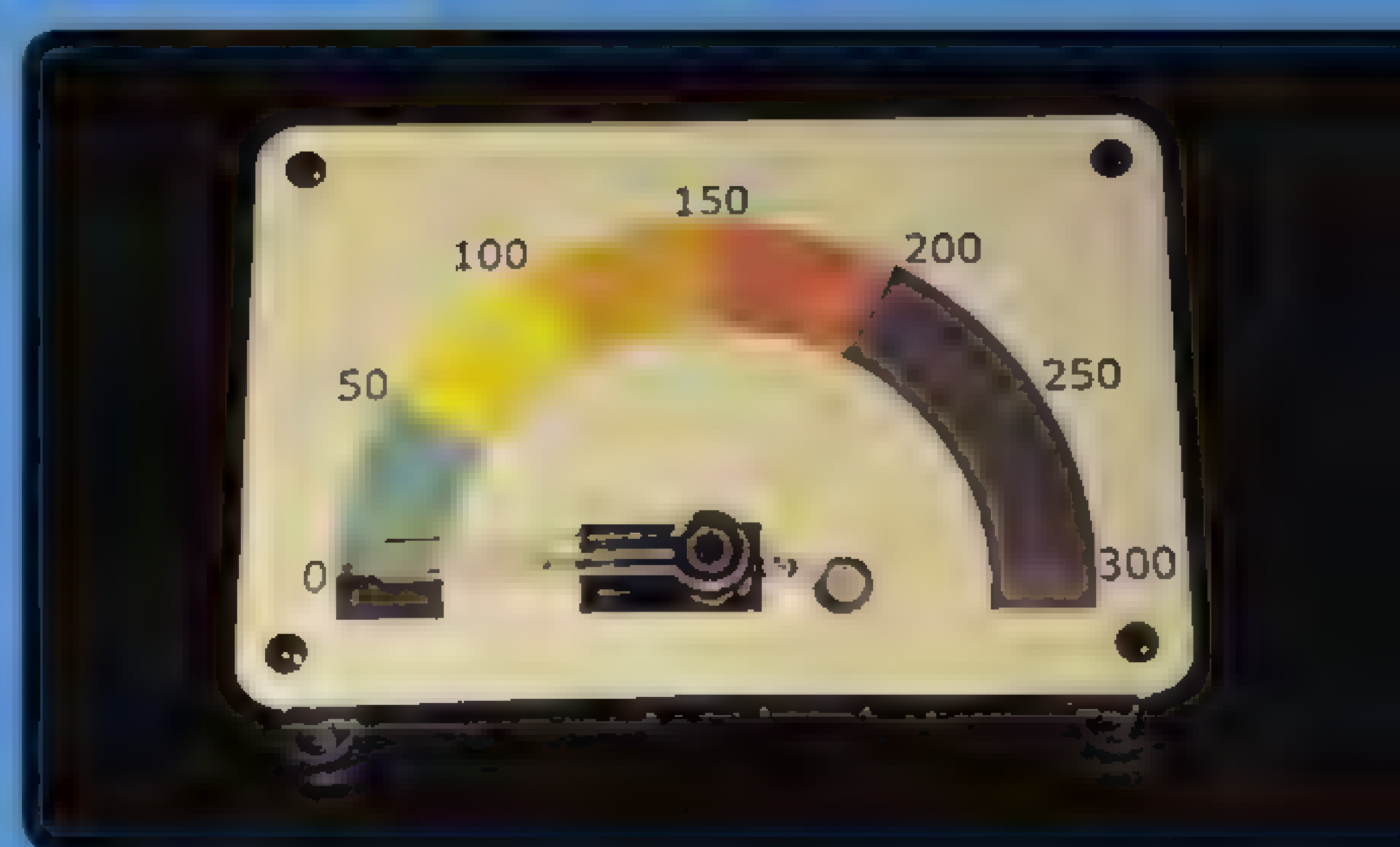
Edgardo Peregrino

Retro Analogue Gauge Air Quality Monitor

Suitability to breathe-o-meter

"In this project, I show you how to track the air quality (using public data from the popular PurpleAir Air Quality Sensors) and displaying it on a retro analogue gauge that shows the colour-coded Air Quality Index (AQI). It is built with a Raspberry Pi, a micro-servo, an RGB LED, and a ProtoStax Enclosure along with some Python."

Sridhar Rajagopal



Pico Scroll

Building the basics

Jesper E. Siig @jespersiig

Plant care robot

Aula-J @Aula_J2018

Software projects

Programming is as much making as soldering

Cat detection system

"Probably the easiest make ever. Pi Zero plus old webcam, MotionEyeOS software, Pushover app, and a Python script from *The MagPi* magazine, and hey presto, a cat detection system up and running #monthofmaking".

Robert Kelso @cumblebob

Clumsy MIDI

"I've been playing with my #mt32pi synth (by @_dopefish_) to see what it can do."

diyelectromusic @diyelectromusic

Internet speed monitor

"In this #MonthOfMaking, I built an internet speed monitor using Docker on a Raspberry Pi to figure out why my internet was so slow. Blog here: **magpi.cc/speedmonitor**."

Daniel Sharp @DanSharpC

Workshop temperature monitor

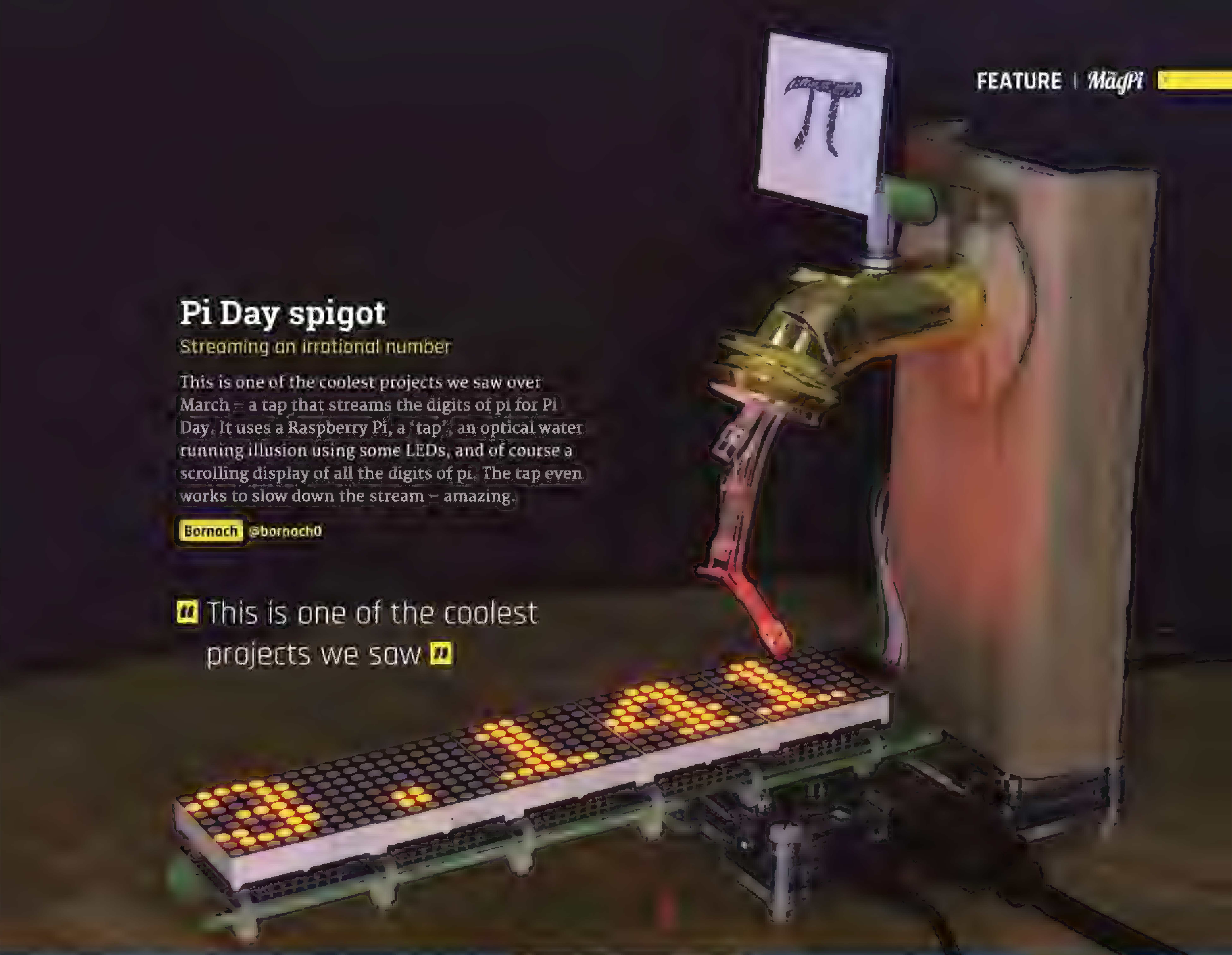
Track your working environment

Brian Corteil @CannonFodder

Dancing robot

No strings attached

D.D.



Pi Day spigot

Streaming an irrational number

This is one of the coolest projects we saw over March – a tap that streams the digits of pi for Pi Day. It uses a Raspberry Pi, a 'tap', an optical water running illusion using some LEDs, and of course a scrolling display of all the digits of pi. The tap even works to slow down the stream – amazing.

Bornach @bornach0

> This is one of the coolest projects we saw

Torvalds

Super-charged robot

Edgardo again, this time with a robot that he's using as a base to make a 'zero code' robot for younger makers to follow along to. We do quite like robots with tank treads, and this one also seems to have a series of hidden sensors for easier automation.

Edgardo Peregrino @sentairidernerd

Custom robot
Dr Footleg built this robot and was featured in issue 41 of HackSpace magazine (**hsmag.cc/41**).

Temperature display

Pico experiments

Similar to Brian's project, this one makes use of a classic seven-segment display to show the temperature readings. Pico can natively read analogue signals such as those from a [illegible] requires some extra components to convert it.

André Clérigo @mrmaster

Raspberry Pi Zero Smart Glasses

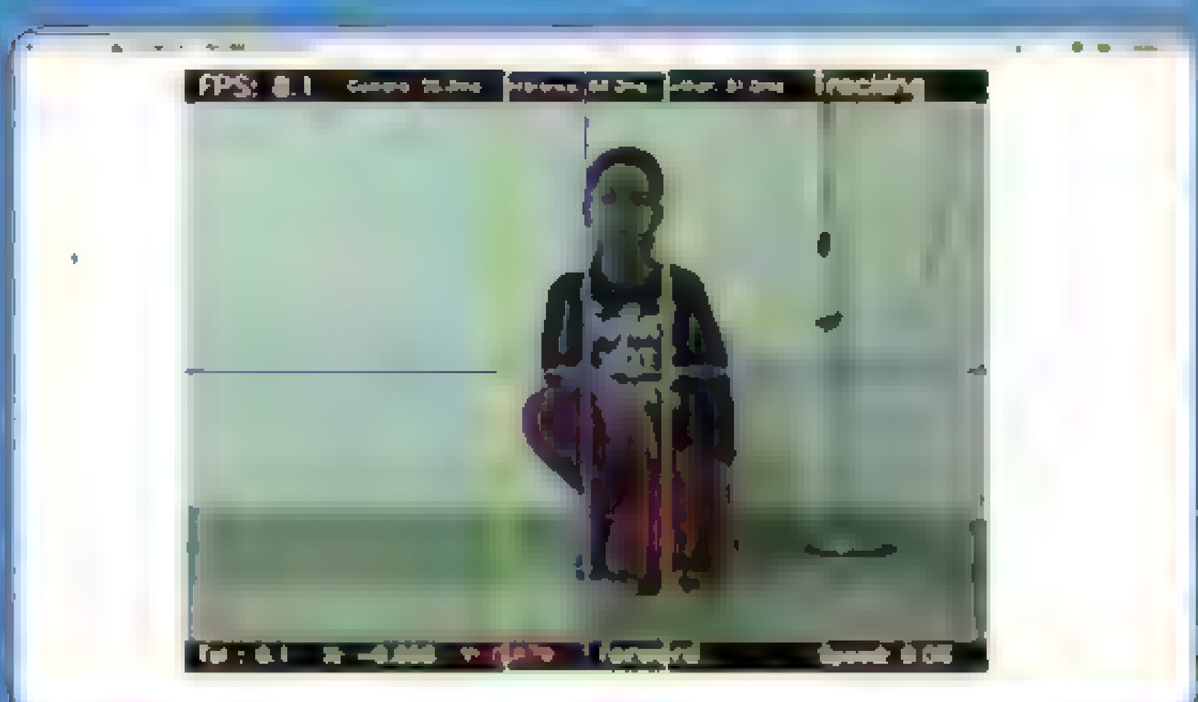
A wearable reborn

Arijit Das

AI Robot

Human-following robot

Jitesh Saini



Alternate makes

Making doesn't just have to be Raspberry Pi-based

Weather frame

"I rebuilt a failed project from 2014 with some LED strips, an Espruino WiFi, a 3D-printed frame, and wooden case. Data comes from OWM API and the six-day forecast is displayed with the colours at the bottom."

Maarten Canmaert
@marzsman

3D-printed art

"3D Printing Art with Fusion360. I wanted to make some decorative blank Eurorack panels and turned to 3D printing to do it."

Liz aka Blitz City DIY
magpi.cc/blitzcitydiy

Watch Liz design and print some cool art and learn how to do it yourself here ***magpi.cc/3dartprint***

Potion of Healing dice shaker

"For #MonthOfMaking I decided to start a whole side business making and selling #DungeonsandDragons gifts on Etsy. I've had some great sales and feedback for these Potions of Healing and can't wait to design more! Each bottle contains four or eight four-sided (d4) dice, with instructions on how many to roll on the label."

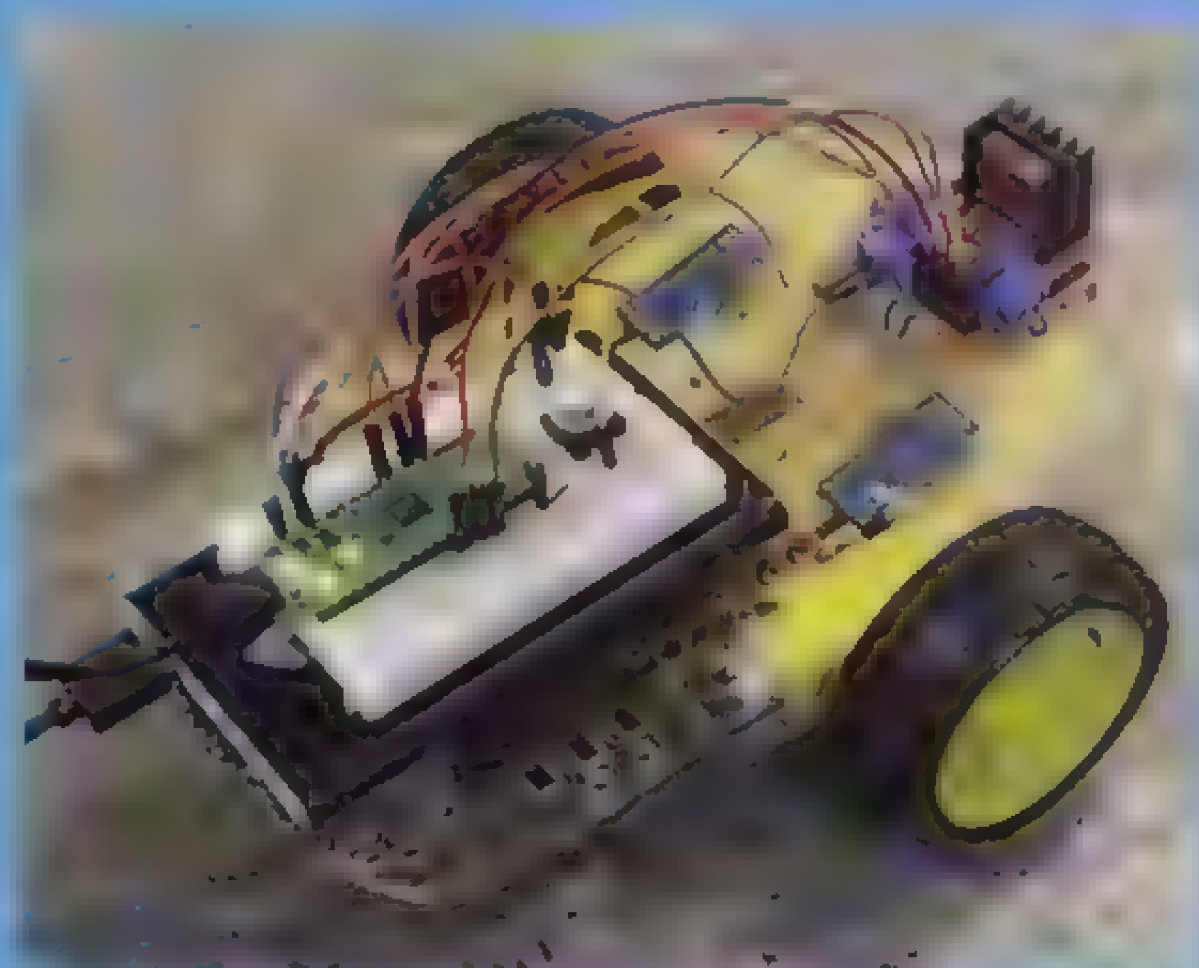
Alex J'rassic @alexjrassic

You can still buy these from Alex's shop here. ***magpi.cc/potiondice***

DIY Desktop Case

Info enclosure

Hexabitz



Pico rover

A simple Pico bot

Robert Wiltshire

HQ Camera mod

Ready for a close-up

This is a simple and clean build that allows for taking great photos with the HQ Camera, and you can preview them on the big 7-inch display as you're shooting. It can even be mounted on a tripod, which is quite fancy.

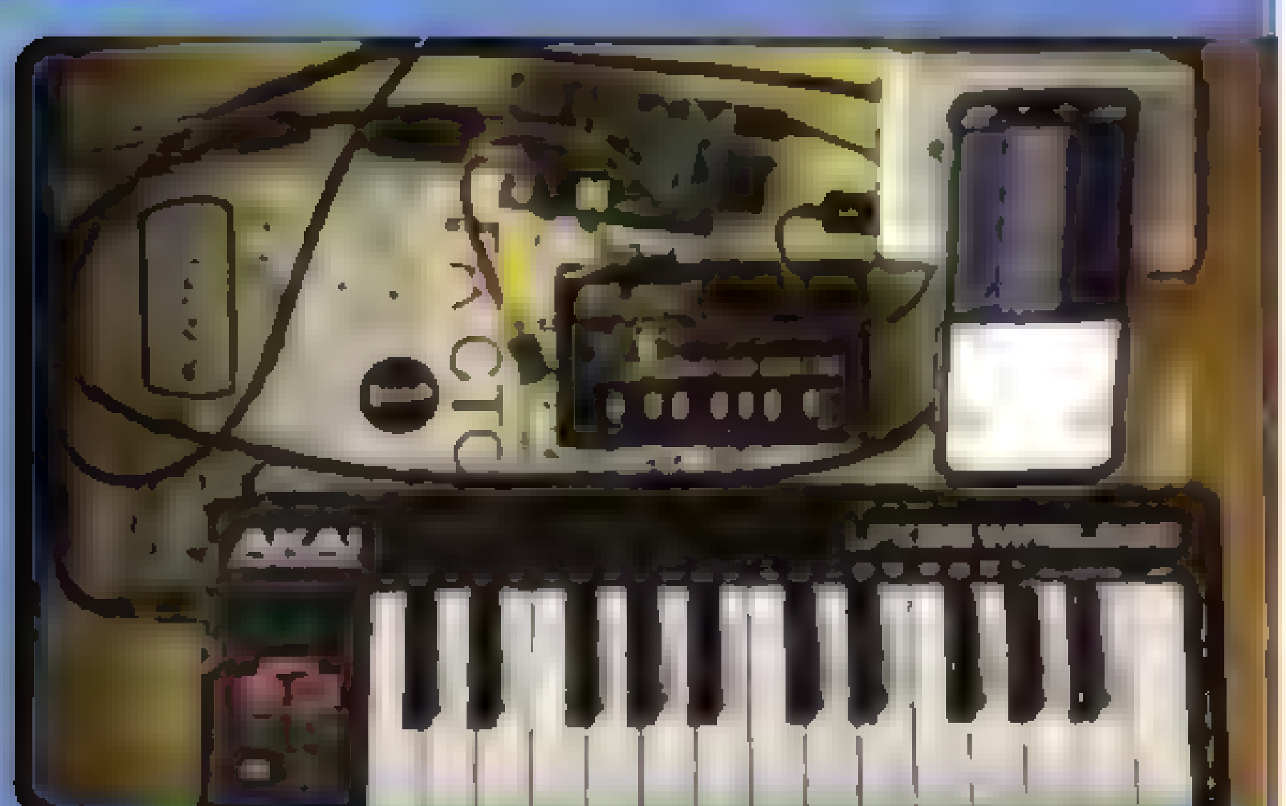
Sridhar Rajagopal

Musical piano stool

Sit on it

We notice an original Raspberry Pi in this setup, showing you can still make very cool things with an older model. Jonny here followed a guide to make his own, so if you fancy, you can too: **magpi.cc/monotron**.

Jonny



Rob's projects

Features editor Rob Zwetsloot made a fair few things over March

RG 1/144 Gundam Exia model

SKILLS: Water decal positioning, airbrushing, toy photography (badly)

I forgot about this kit for two years but it's finally done, and looks lovely!

Souffle pancakes

SKILLS: Baking, folding egg whites, wrestling with bad oven

Souffle pancakes are all the rage – I attempted Ann Reardon's oven recipe with mixed results. More practice!

Raspberry pie

SKILLS: Baking, making jam, pastry lattice, restraint from eating it all at once

I baked this for Pi Day, and ate it on the Raspberry Pi Digital Making at Home Pi Day stream

Live2D rigging for Vtuber

SKILLS: Animation, face tracking, hair physics

Made for Maids of England's April Fool's Day video using art by Tea and Paintwater **(magpi.cc/teanpaint)**

PiDay Pinball

Virtual flipping on Raspberry Pi

Virtual pinball has slowly been gaining traction over the years – VR pinball is just becoming a [illegible] TV that uses a Raspberry Pi to power it.

RzR @RzrFreeFr

> "You can still make very cool things with an older Raspberry Pi"

DIY monitor

Practical upcycling

Rafa sent this one to us via email – he managed to use an old notebook as a screen for his Raspberry Pi, saving a lot of money in the [illegible] [illegible] Raspberry Pi as a main PC for a year."

Rafa Coringa

Grove Starter Kit for Raspberry Pi Pico

▶ Grove ▶ **magpi.cc/grovestarter** ▶ £TBA / $43

Explore the world of Raspberry Pi Pico electronics with this click-and-play kit. By **Lucy Hattersley**

SPECS

SIZE: 56 mm × 56 mm

PORTS: 3 × analogue, 3 × digital, 2 × UART, 2 × I2C, 1 × SPI

OPERATING VOLTAGE: 3V3 / 5V

Exploring the world of electronics is one of the fundamental joys of using a microcontroller such as Raspberry Pi Pico (magpi.cc/pico). The GPIO pins on Pico can be wired up to an array of sensors, buttons, LEDs, and displays enabling a wide array of projects.

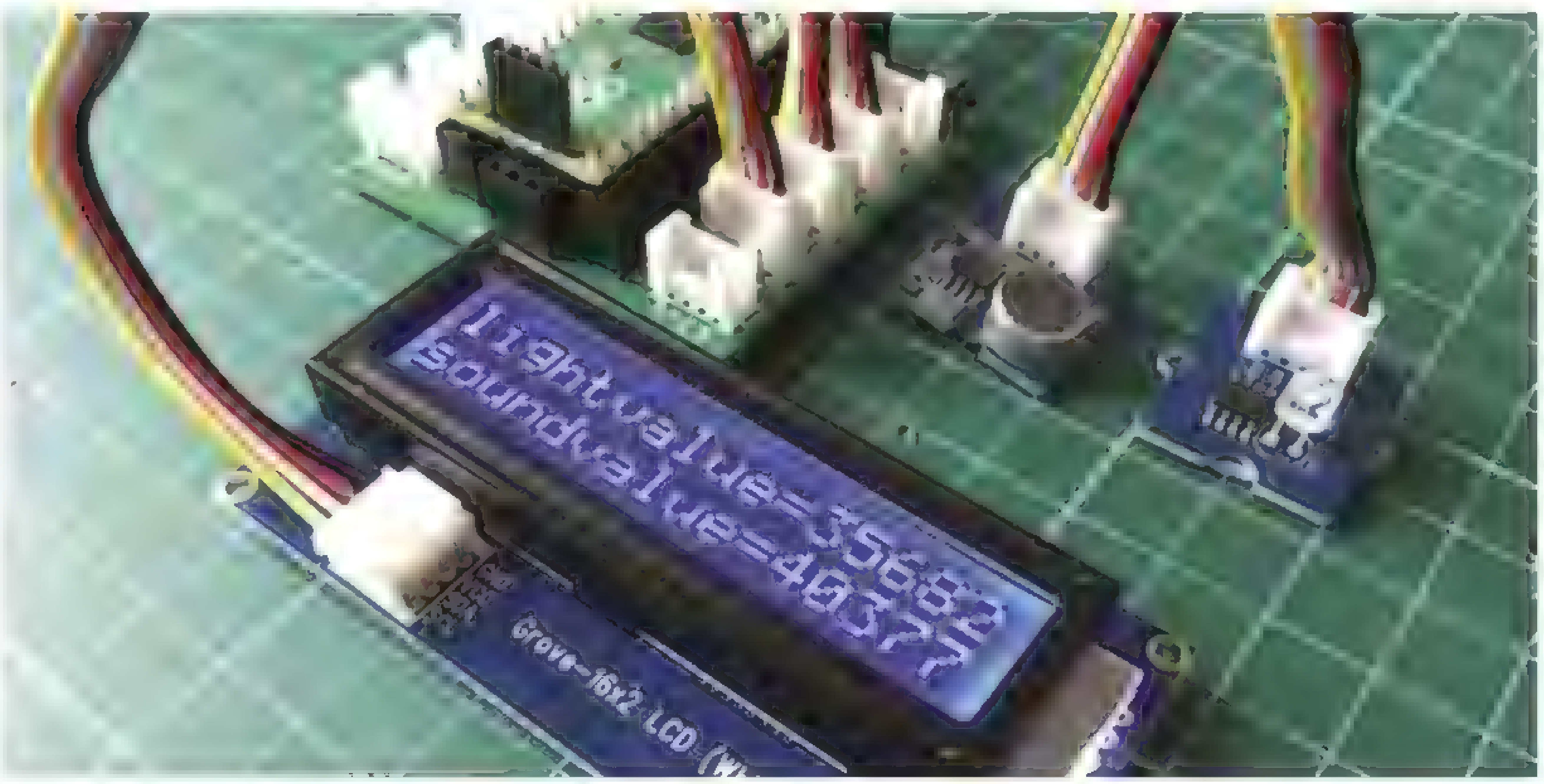
At the heart of the Grove Starter Kit for Raspberry Pi Pico is a Grove Shield ($4, **magpi.cc/groveshield**). This board has ten different Grove Connectors: three analogue ports, two digital ports, three UART ports, and four I2C ports. It also has a 3V/5V power switch to adjust the voltage output, and a six-pin SPI interface.

Raspberry Pi Pico slots into the header on the Grove Shield and Grove parts are snapped straight into the white Grove connectors. This enables you to prototype projects quickly and easily.

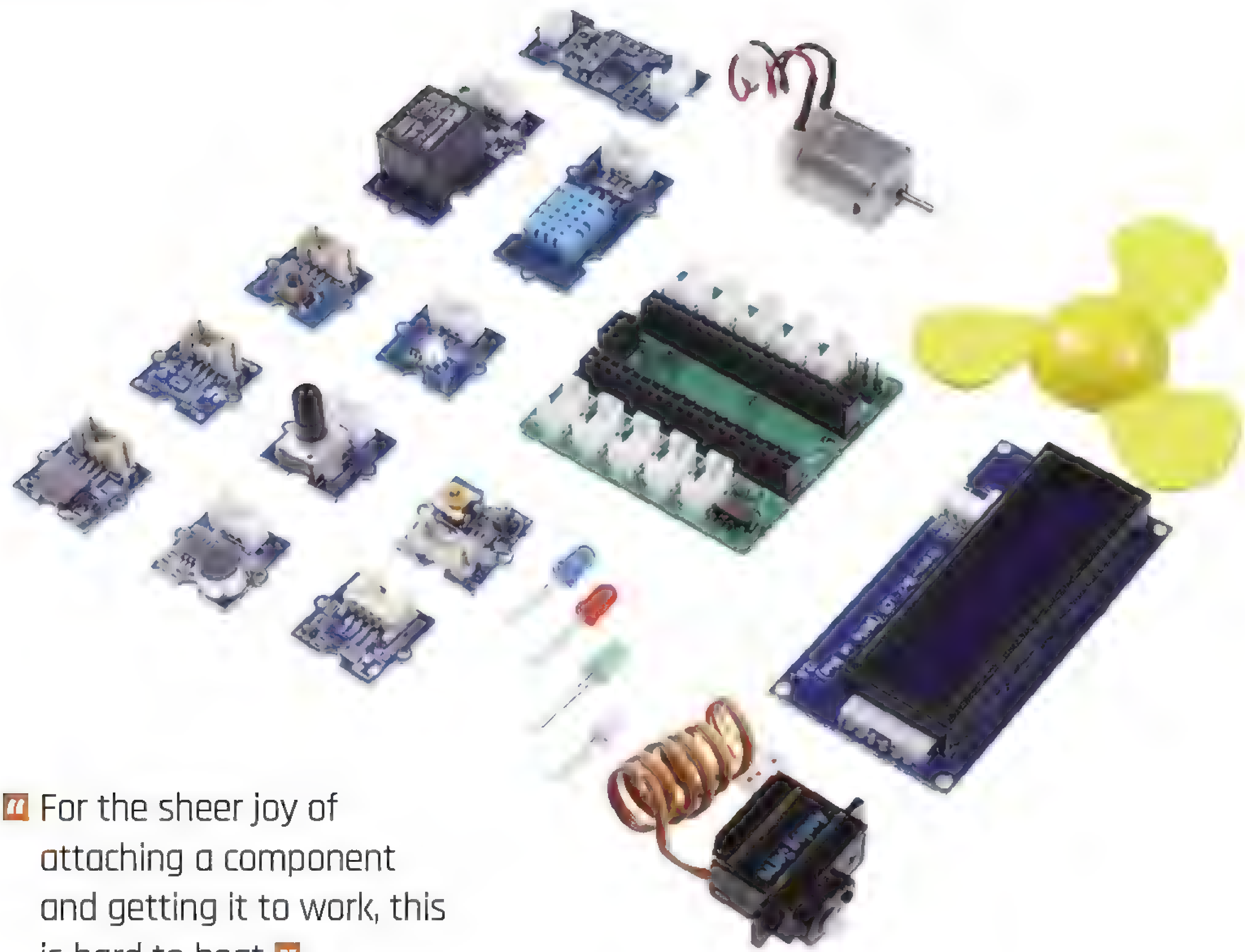
You will need to have soldered pins on to your Pico (or you can pick up a pre-soldered Pico, **magpi.cc/picosoldered**). But from that point on, you don't need to solder parts or figure out circuits and jumper wires. You just connect the Grove part to the Grove Shield using one of the included Grove Universal 4-Pin cables (**magpi.cc/grovecable**). Grove itself says this "simplifies the learning system, but not to the point where it becomes dumbed down".

Each Grove cable has four wires: typically one for power, another for ground, and two for input and output (the exact nature of each wire depends on the part it's connected to; you can read more at **magpi.cc/groveinterface**).

This does, indeed, make it extremely easy to hook up components to Raspberry Pi Pico. And to that



▶ The Grove Shield connected to a 16×2 LCD and two sensors. Here the display is programmed to output light and sound values



> For the sheer joy of attaching a component and getting it to work, this is hard to beat

end, the kit comes with a wide range of parts to play with. There is an LED pack, RGB LED display, light sensor, sound sensor, rotary angle sensor, temperature and humidity sensor, passive buzzer, button, servo, mini fan (with DC motor), relay, and a 16×2 character LCD.

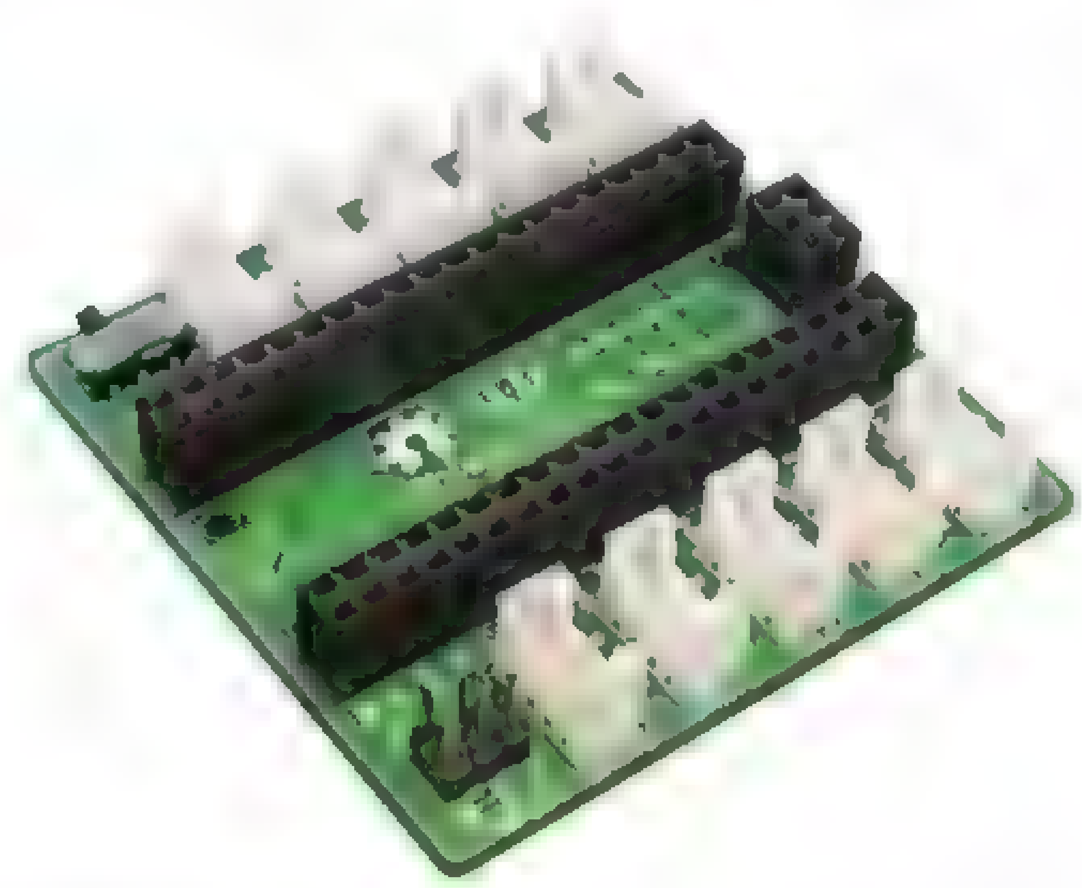
Learning curve

The Grove Shield for Pi Pico wiki page has a range of tutorials that use the parts found in the kit (**magpi.cc/groveshieldwiki**).

The tutorials use all the parts found in the Starter Pack and give a good overview of what you can do. You typically need to download a Python module for each part. And analysing the code will give you a good overview of what each component can do. We programmed the LCD to respond to light and sound; a fan and servo movement detector; and played around with lights, buttons, and the relay.

Thanks to the Python support files, introductory wiki tutorials, and the click-and-play nature of the kit, it is ridiculously easy to move from having an idea to getting it working.

There is an argument that replacing the pure jumper wires and breadboard with a custom connector removes part of the learning curve. And it's hard to take a prototyped circuit and wire components directly to Pico so you remain attached to the Grove Shield and its ecosystem of parts. But, for the sheer joy of attaching a component and getting it to work, this is hard to beat.



▲ The Grove Shield for Raspberry Pi Pico has a range of ports that make it easy to click and connect components

▲ The Grove Starter Kit contains a range of Grove components and the Grove Shield

Verdict

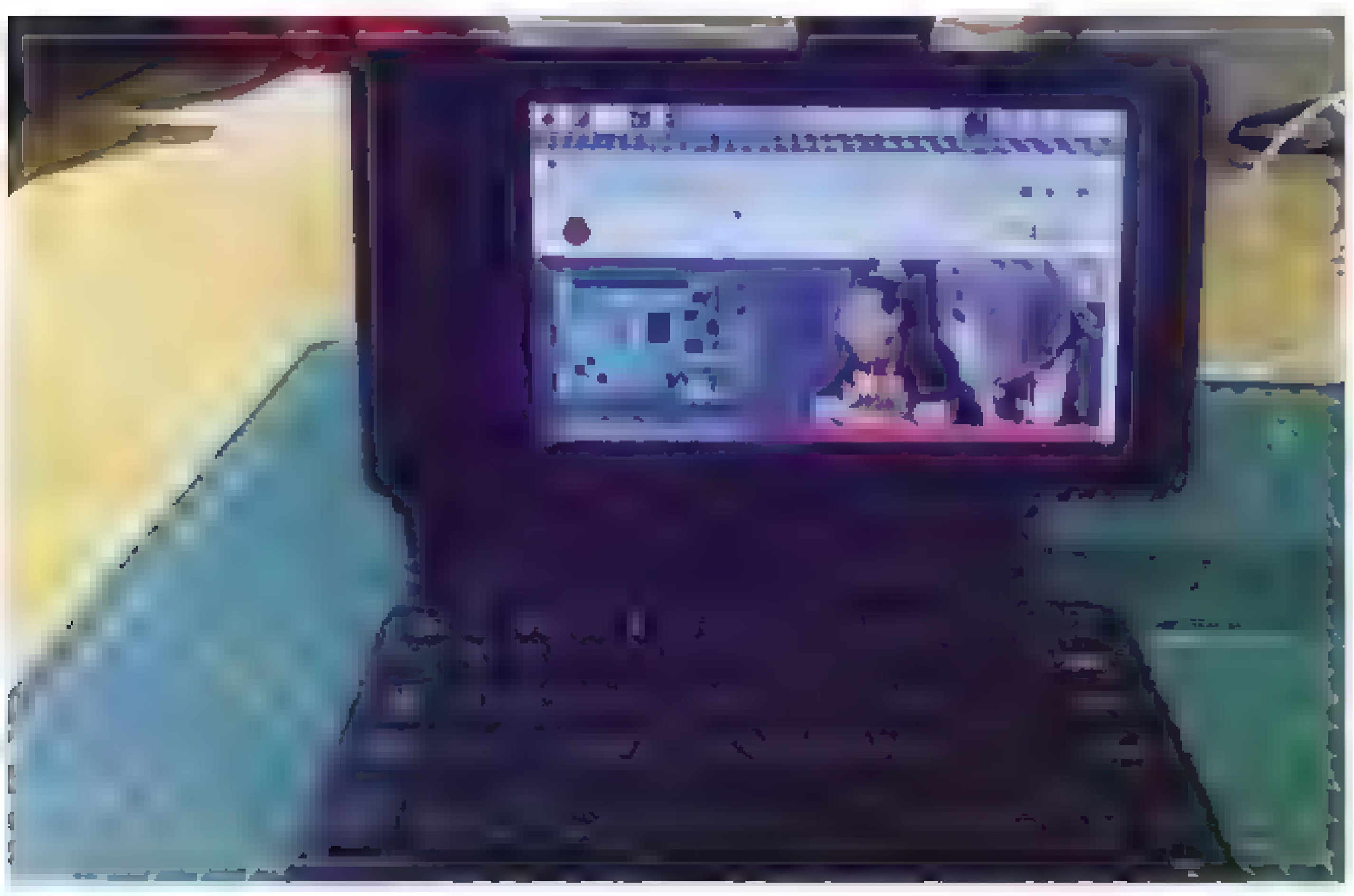
It offers a painless way to attach components to Raspberry Pi Pico and play around with electronics. There's a huge range of components, and the documentation is good.

9/10

10 Amazing: Portable Raspberry Pi projects

Take your Raspberry Pi with you for extra fun

Raspberry Pi is small and requires very little power. This makes it perfect for plugging in a portable battery and making your trips a bit more interesting. Here are some of the greatest ways to make your Raspberry Pi portable.



▲ Raspberry Pi 2Go portable workstation

Making-to-go

This compact design is great for making and testing on the go, especially with the built-in breadboard!

magpi.cc/98

▲ Ruggedized Raspberry Pi Portable

Apocalypse-ready

When planning for many kinds of fictional apocalypse, people seem to always forget the tech part. This rugged laptop will help there *and* in the real world.

magpi.cc/ruggedpi

◂ PiBoy DMG

Game Boy converter

There are a few ways to turn a Raspberry Pi into a portable handheld, but this all-in-one kit from PiBoy DMG is one of our recent favourites.

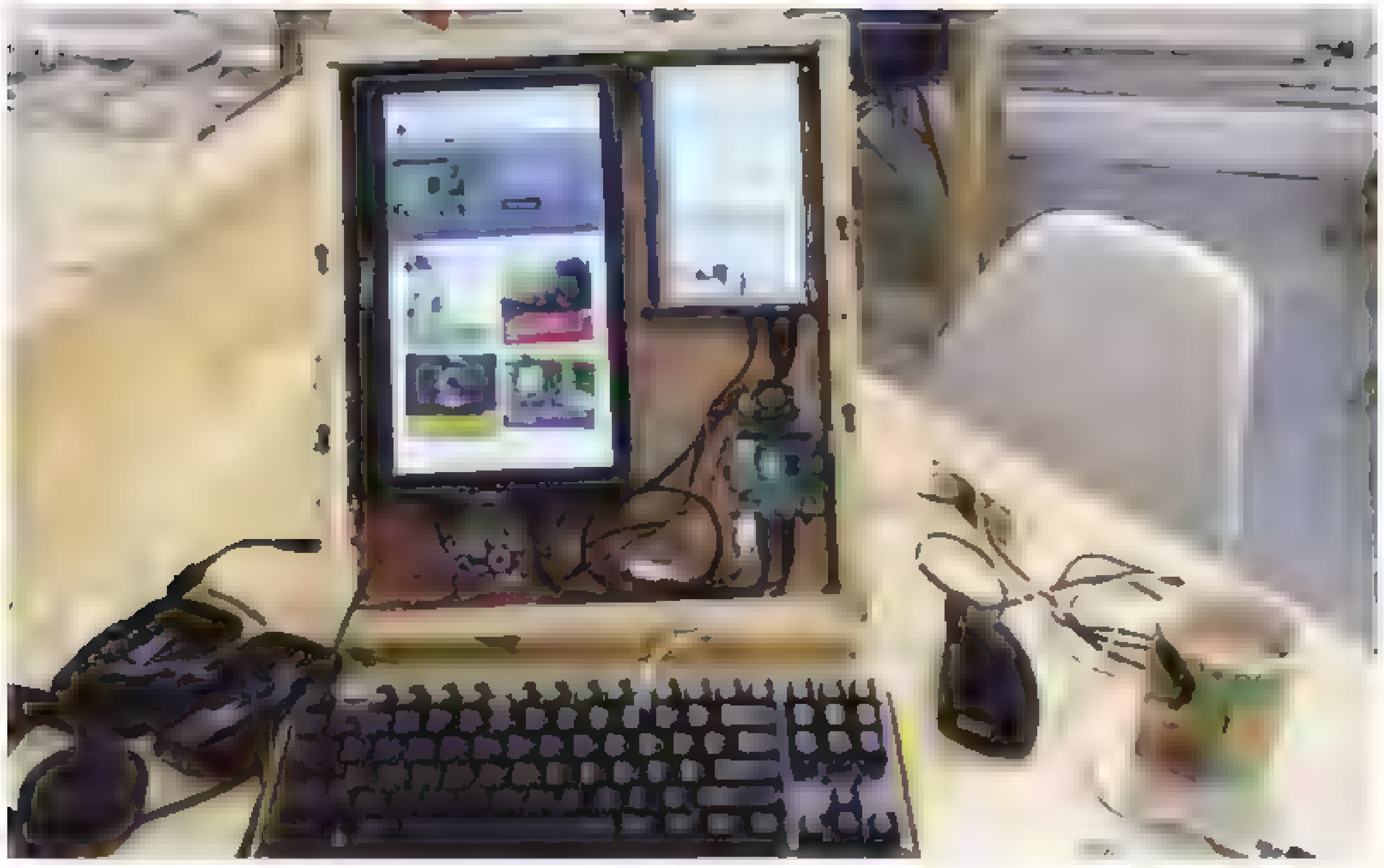
$130 | magpi.cc/piboydmg

▲ Lunchbox Arcade Game

Need food badly

A quick go at Street Fighter on a lunch-break is a tradition in some offices, and this takes it a little step further by combining the two.

magpi.cc/lunchbox



▲ Portable Raspberry Pi 4 computer

Recycled parts

HackSpace magazine editor Ben Everard made this very quickly with some spare parts and a handmade wooden 'case'. It does the job surprisingly well.

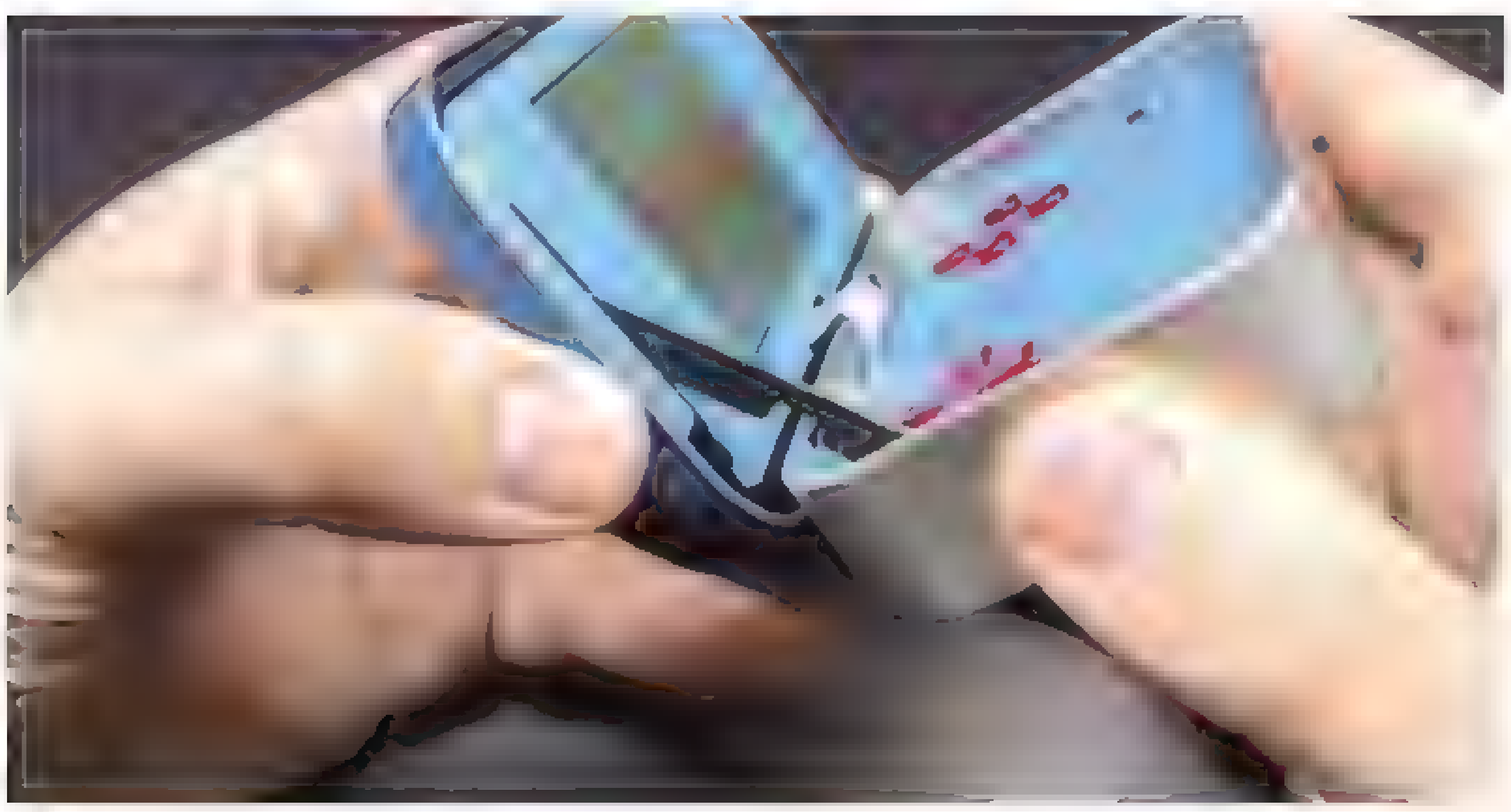
hsmag.cc/24

▸ Wearable time-lapse camera

Conspicuous fast-motion

A very cool use of Raspberry Pi that lets you walk around and get a nice little time-lapse of your day.

magpi.cc/timelapse



▲ mintyPi

Incognito gaming

Stuffing a Raspberry Pi Zero into a mints tin is an incredible accomplishment in itself, and mintyPi takes it a step further to be a whole game system.

magpi.cc/mintypi

▲ Raspberry Pi laptop

Pocket power

This tiny Raspberry Pi laptop from issue 74 is the whole package in one – there's even a rechargeable built-in battery!

magpi.cc/74

▲ Becca Cam

HQ Camera hack

The recent HQ Camera is a powerful sensor with some great lenses. It's not a full camera, though, so Becca put it inside an old SLR.

magpi.cc/beccacam

▸ Raspberry Pi Zero AirPlay

Music anywhere

Need a portable speaker that anyone can control easily? This AirPlay speaker is simple and looks great.

magpi.cc/airplay

Learn web development with Raspberry Pi

Develop websites and internet-connected services with Raspberry Pi. By **Lucy Hattersley**

HTML & CSS Modules 1 & 2

AUTHOR

Code Club

Price
Free

magpi.cc/webdev1

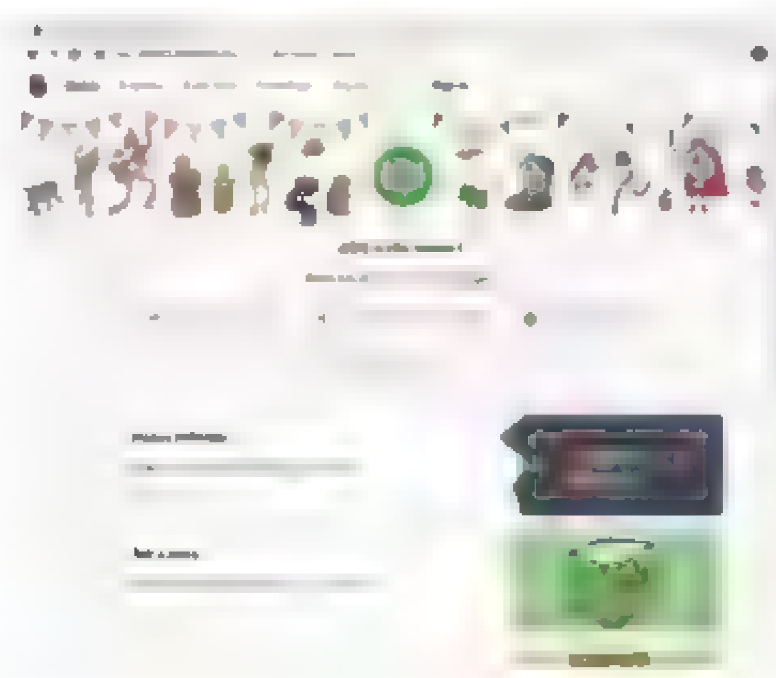
Web design, and web development, are great skills to have. Web design enables you to create websites; web development enables you to build the back-end server and integrate it with the many hardware and software features of Raspberry Pi.

Many builds and coding projects benefit from having a good website (either as part of the build, or as an accompanying help and documentation site).

The good news is that web design is – at least in the early stages – remarkably easy to pick up. Although, like many skills, it can take a lifetime to master.

Absolute beginners should check out these modules made by Code Club. They walk you through HTML and CSS (the text and style code used at the heart of website design). Designed to keep kids interested, these lessons follow the form of a practical project, such as designing a wanted poster or displaying a recipe. Later lessons work up to creating an interactive pixel-art editor and animated graphics.

The code is displayed and coded in an interactive online environment called Trinket (**trinket.io**). You can follow along with the web browser on your Raspberry Pi.



Web development books

Flick through these paper or PDF books

WEB DESIGN WITH HTML & CSS

Jon Duckett's HTML and CSS book is a popular guide to web development. The books are highly visual, with artwork and photography being used to explain the concepts. A joy to read.
magpi.cc/duckett

RASPBERRY PI FULL STACK

Dr Peter Dalmaris's guide to using Raspberry Pi is over 400 pages long. It goes over setting up Raspberry Pi and an NGINX stack, before moving on to data logging, charts, integration with database, and setting up a Raspberry Pi with Arduino and sensors.
magpi.cc/elektorfullstack

FULL STACK WEB DEVELOPMENT WITH RASPBERRY PI 3

It's a little older, but Soham Kamani's book is still relevant, and covers setting up a network stack and running a Node server. Then you can extract data from the GPIO pins and sensors, and use this to create web pages displaying the data.
magpi.cc/packtfullstack

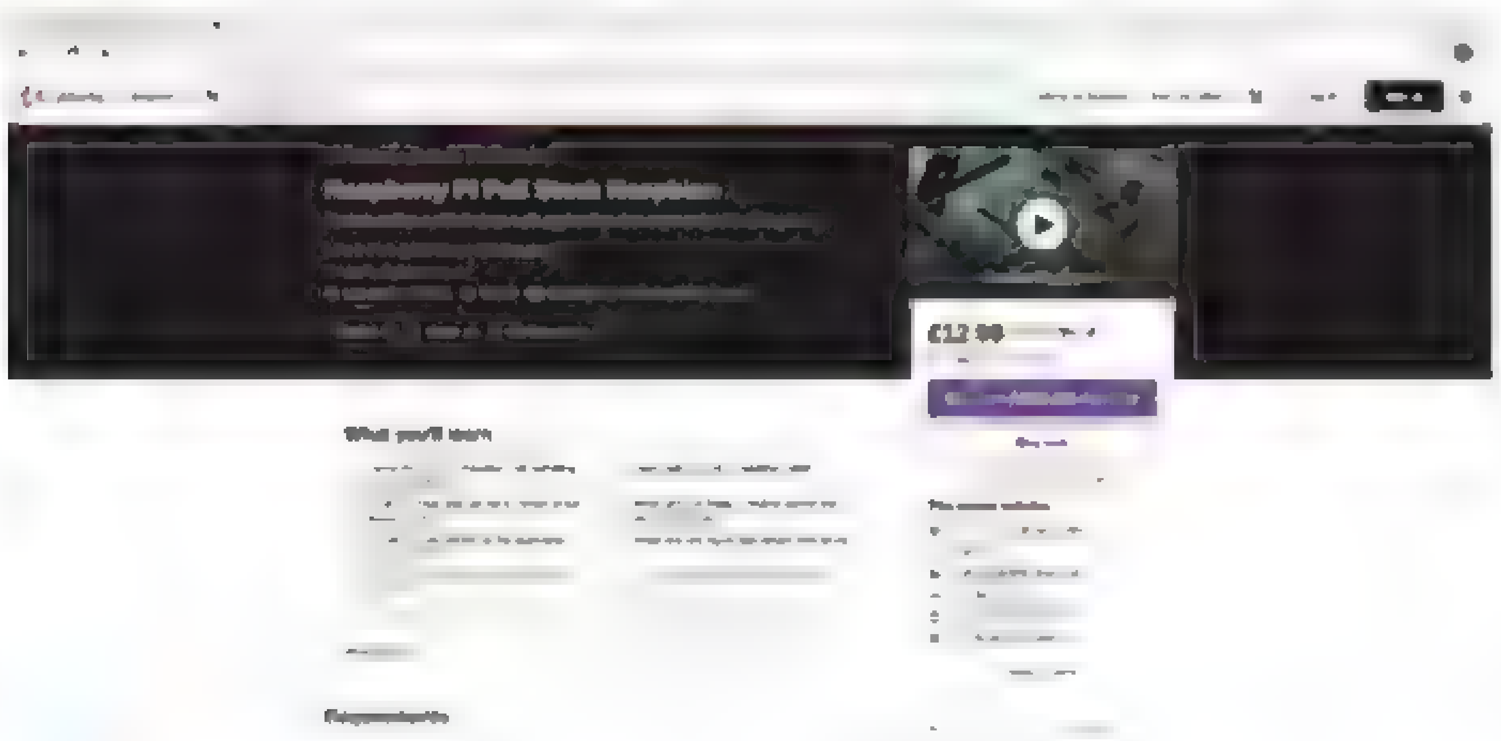
Raspberry Pi Full Stack

Udemy

Price
£60/$60

magpi.cc/udemyfullstack

Once you have the basics of HTML and CSS, it is time to get serious about web development. Few courses offer the breadth and depth of Udemy's Full Stack for Raspberry Pi. This hands-on project is based on building an Internet of Things application.



The course uses a DHT22 sensor (**magpi.cc/dht22**) with an LED, push-button, and breadboard. With those in hand, you set up Raspberry Pi as an NGINX web server and SQLite database, and use HTML and JavaScript to control everything.

It's far more detailed than most web development courses, and it is built on top of Raspberry Pi hardware, which makes it ideal for our needs. The course is paid for, but you do get nine hours of video and a range of code resources. It's frequently on sale too, and keep an eye out for vouchers (it's currently on sale for £13 in the UK).

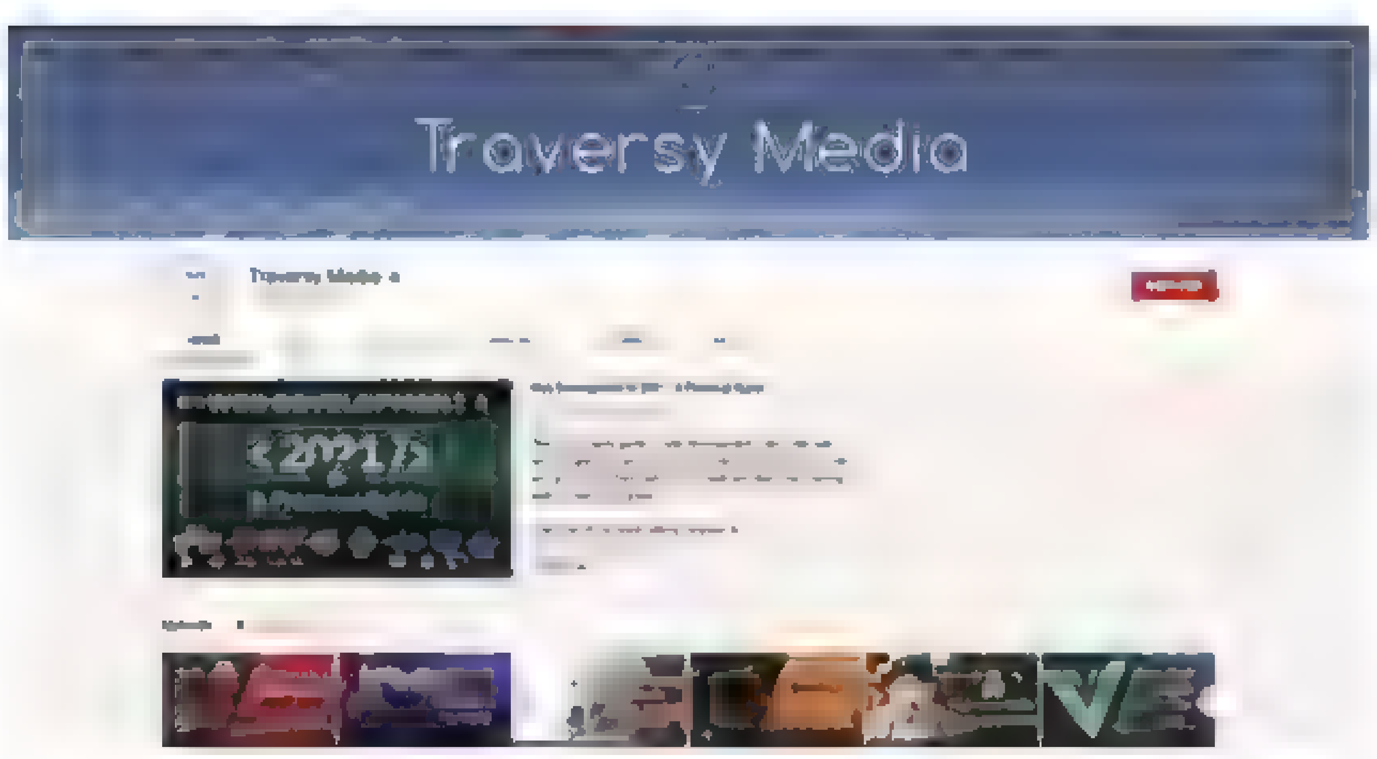
Traversy Web Development

Brad Traversy

Price
Free

magpi.cc/traversy

If you like to learn web development by watching YouTube, then Brad Traversy has one of the biggest channels around. There are videos on all the main subjects, including crash courses in HTML, CSS, and JavaScript. Plus detailed videos on various stacks (such as React & Django and MERN). And there are web servers and database technologies. If that's not enough, Brad goes into other technologies such as Docker, Python (including Python Flask for web app development), WordPress theme development, and even general subjects such as career and personal help. One video, titled 'Overwhelmed with web development technology', seems apt.



There is a lot to digest here, and it's all free to watch and generally enjoyable to sit through. Although interactive courses offer a greater learning experience than YouTube video, so it's good to combine this resource with a tutorial or book. Brad offers a range of Udemy courses to sign up for on his website (**traversymedia.com**). These courses offer more detail than the YouTube videos, and you can learn as you work through the projects.

Bookmark these websites

There's no shortage of websites about web development

FREECODECAMP

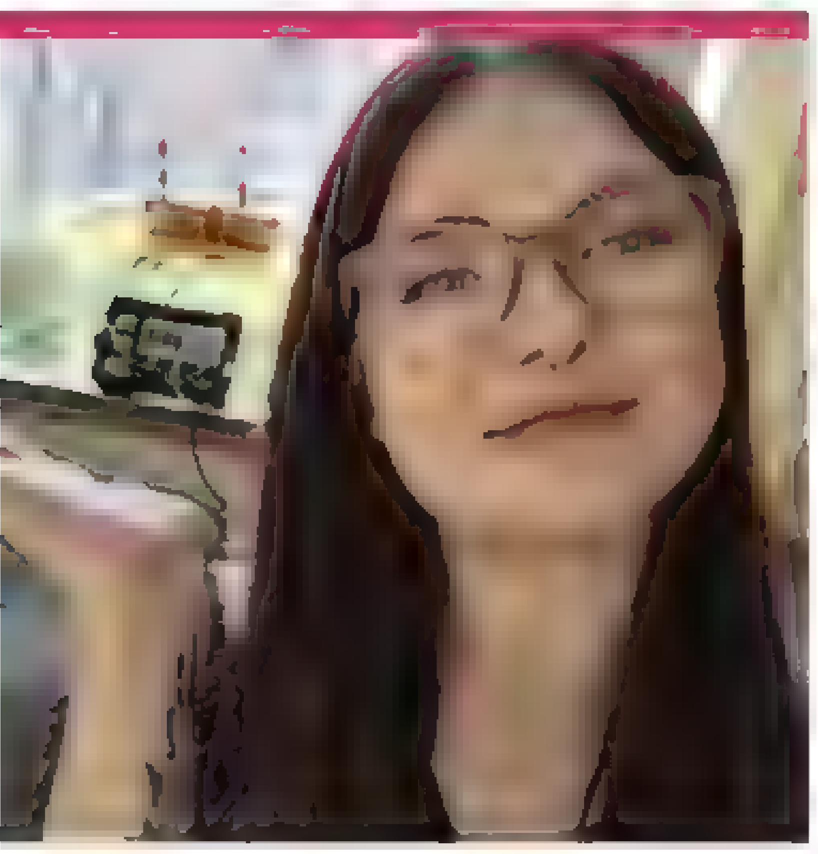
There are lots of websites offering great content at cost, but freeCodeCamp does it all for free. It covers every web development subject under the sun and offers certifications.
magpi.cc/freecodecamp

GETTING STARTED WITH THE WEB

Getting started with the Web is a series of tutorials from Mozilla that covers HTML, CSS, and (as the name suggests) all the basics. It's a clean guide to getting started and a great resource to bookmark.
magpi.cc/mozillagettingstarted

COURSERA

Another free offering with certification. Coursera has a wide range of web development courses offered by universities. University of Michigan's 'Web Design for Everybody' and Johns Hopkins University's 'HTML, CSS, and JavaScript' are highly rated.
magpi.cc/coursera



Ellora James

Meet the smart young computer scientist behind an exciting new YouTube channel

- Name **Ellora James** | Occupation **Student**
- Community role **YouTuber** | URL **magpi.cc/ellora**

When did you have that 'ah-hah' moment with computing? Ellora James remembers when she had hers.

"When I was about 14/15 years old, I was considering taking Computer Science at exam level, and so my teacher gave me a Raspberry Pi to borrow and a booklet on Python to play about with," Ellora tells us. "I remember taking it all home, setting up Raspberry Pi on my kitchen table, and writing my first-ever line of python where I got it to print 'Hello World'. I was just fascinated by the concept of being able to get computers to do what we tell them, and I've been hooked ever since."

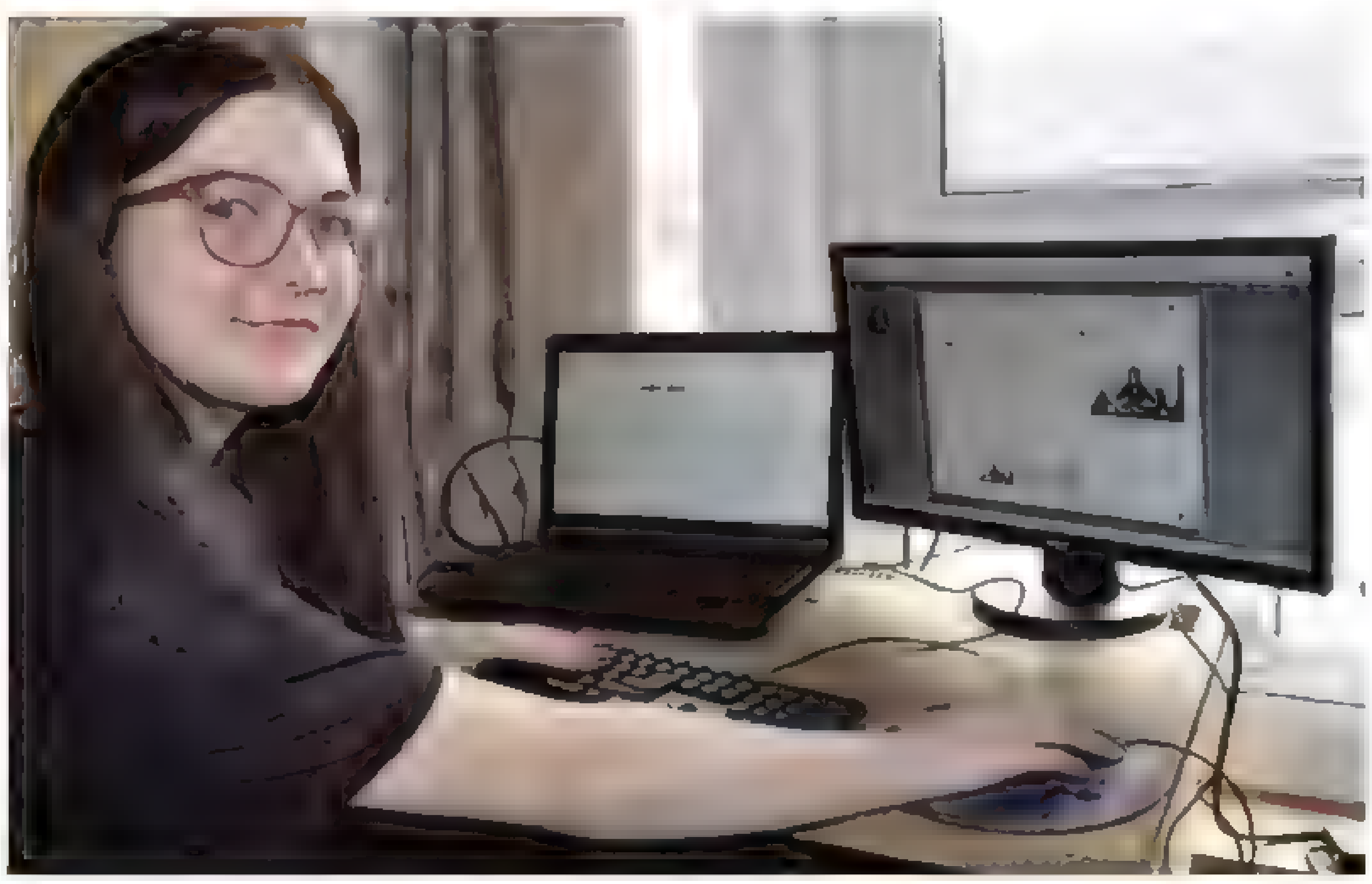
Several years, one company, and many awards later, Ellora is an up-and-coming young computer scientist who has recently launched her own amazing YouTube channel.

What is your history with Raspberry Pi?

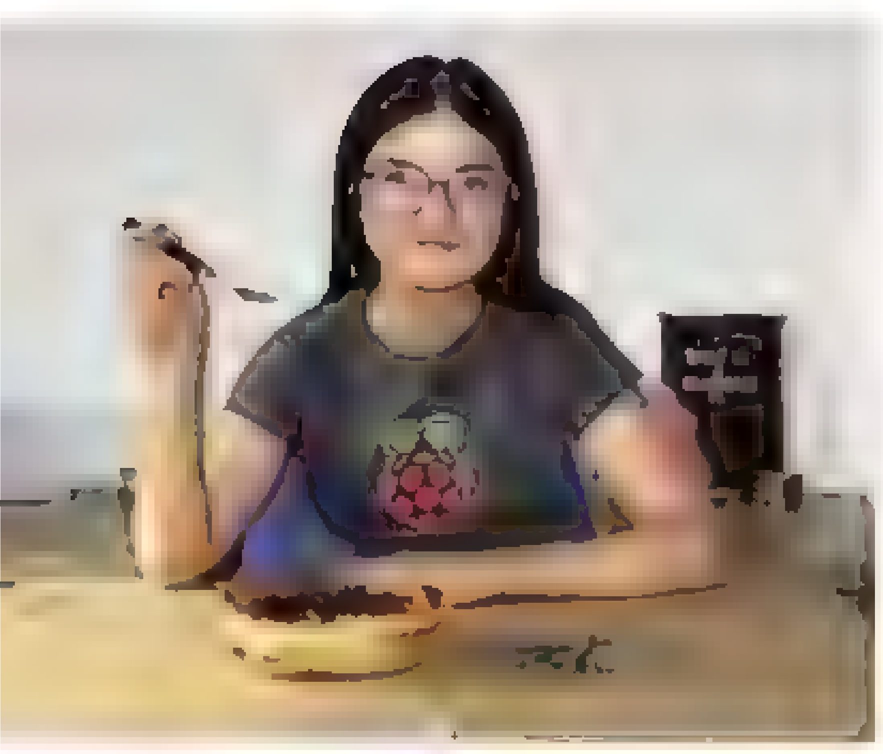
I continued to work through the Python booklet at home and started to dive deeper into Raspberry Pi itself. The big thing that really cemented my love of Raspberry Pi was being selected to attend the Raspberry Pi Digital Making Day in Cambridge, back in 2016. We had to create a 60-second video to apply and despite mine, from what I can remember, being very cheesy, I was selected to attend. Me and my Mum travelled to Cambridge, which was a fun trip in itself. We tried soldering, set up bird-boxes with infrared cameras and Raspberry Pi Zero, and got to use electric paint to create circuits on T-shirts. It was so much fun, and I'm still planning on setting up my bird-box one day. Being specifically selected to attend gave me a lot of motivation to keep making. And that must have stayed with me over the years, as I'm back making again, and this time on an even bigger scale with my channel!

What is Envirocache?

Envirocache is a mobile app designed to get children (and adults) outside, active, and educated about the world around them. The app allows you to



Ellora manages her YouTube channel mostly by herself which can be a lot of work



Our favourite pie and microcomputer in one shot

Ellora's new YouTube channel includes fun builds with Raspberry Pi

search for walking routes near you and shows you points of interest you can find along the way. You unlock badges and earn points for finding these, like a nature treasure hunt. The concept started off as an entry to

> The concept started off as an entry to the Apps for Good competition

the Apps for Good competition and has grown from there. I now work on the app alongside my team members, Mari-Ann Ganson and Jamie Smith. The app is still in its development stage, but we're looking at a potential release date this year.

What are some of your favourite things you've made with Raspberry Pi?

My first big project was my 'PiAlarm'. I got a touchscreen display for the Raspberry Pi for Christmas back in 2015, and ended up designing an alarm clock in Python. It took many hours of work and I also lost my code at one point and had to start again, but I learned a lot from that project, and being my first big one it definitely holds a special place in my heart. It had a full user interface, let me pick from multiple songs, and even tweeted the time it took me to get up and turn it off in an effort to encourage me to get out of bed quicker. I actually made a short video about it and still have the code somewhere, so I'm thinking of revisiting it on my channel as a future project. I also took part in the 2016/17 European Astro Pi Challenge, where my project idea involved measuring environmental data on the ISS to see how this impacted the circadian rhythms of astronauts. The idea was that the astronauts would log how long they slept for, the quality of their sleep, and also hunger levels, to gain an understanding of their circadian rhythm. I also planned to create a reaction game to test how sleep and hunger levels affected their reaction times.

The Grow HAT is part of a plant automation video

One of Ellora's first projects was the PiAlarm. A simple but important build for her

This Month in Raspberry Pi

AI-suke the robot

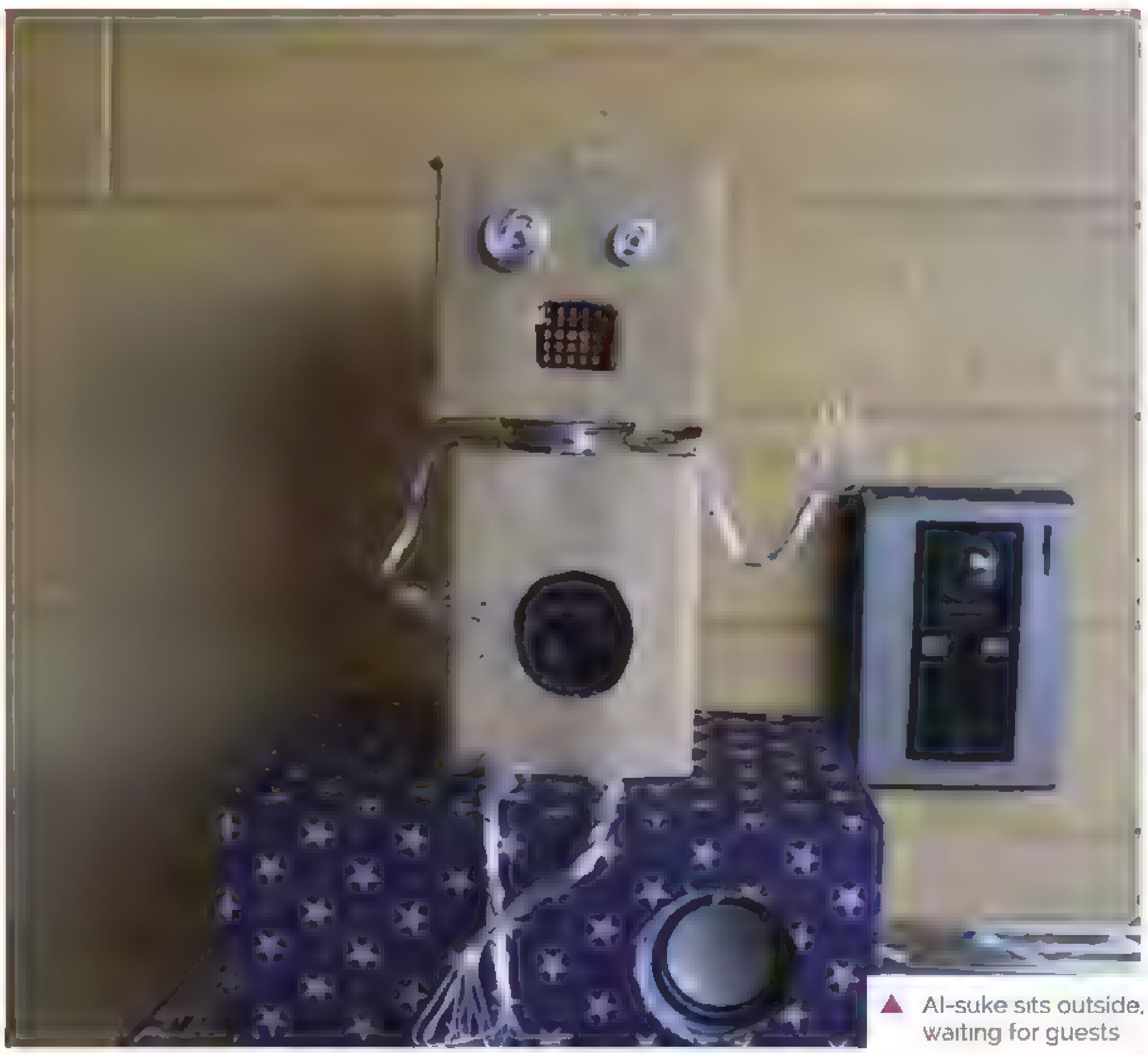
This amazing robot, developed by a young maker, helps greet people

In Japan, there's a yearly programming contest for young makers organised by the PCN (Programming Club Network). PCN helps kids learn to code, so in some ways it's similar to CoderDojo and its Coolest Projects.

This year's winner, Rito Takahashi, created AI-suke (I-sue-kay), a robot that uses facial recognition to greet people at the door. In a time when people are staying at home, it helps to avoid unnecessary contact.

"I named the robot AI-suke, which means 'AI help'," Rito tells us. "AI-suke has a built-in camera and speaker, and faces and voice messages are registered in advance. AI-suke can do face recognition to identify the person who visits the house, and play a voice message tailored to that person."

It will also send notifications to LINE, a smartphone instant messaging service, so you can know if someone has arrived while you're out of the house.



▲ AI-suke sits outside, waiting for guests

Design and success

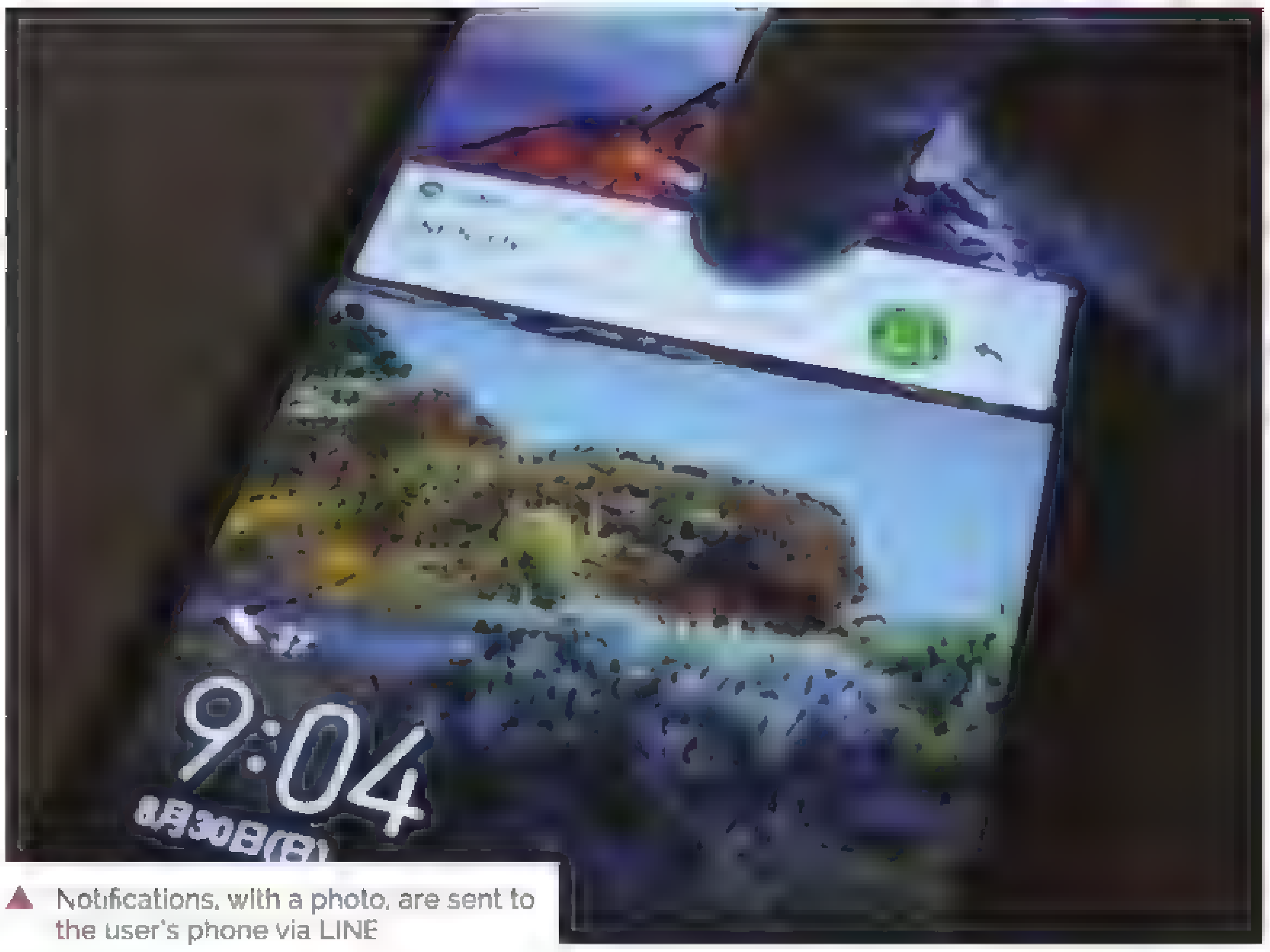
AI-suke has a 'face', as well as other humanoid features, which gets people to look at it. This, in turn, means their face can actually be seen by the camera.

"The people talking to AI-suke feel calm and happy because of its cute voice and appearance," Rito says. "My five-year-old brother acted as the voice actor. The cute body is made with cardboard and wire. These give people a warm impression, as well as ease the anxiety."

You can view a video of AI-suke in action here: **magpi.cc/aisuke**. It's in Japanese, but you'll get the idea.

What does the future hold for this young maker?

"I would like to make things that are not only useful for people, but also make people happy. Raspberry Pi has many powerful functions, so I will try them to improve my future work."



▲ Notifications, with a photo, are sent to the user's phone via LINE

▼ Visitors see AI-suke and look at it. An automated, pre-recorded response is played depending on who is there

MagPi Monday

Amazing projects direct from our Twitter!

Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

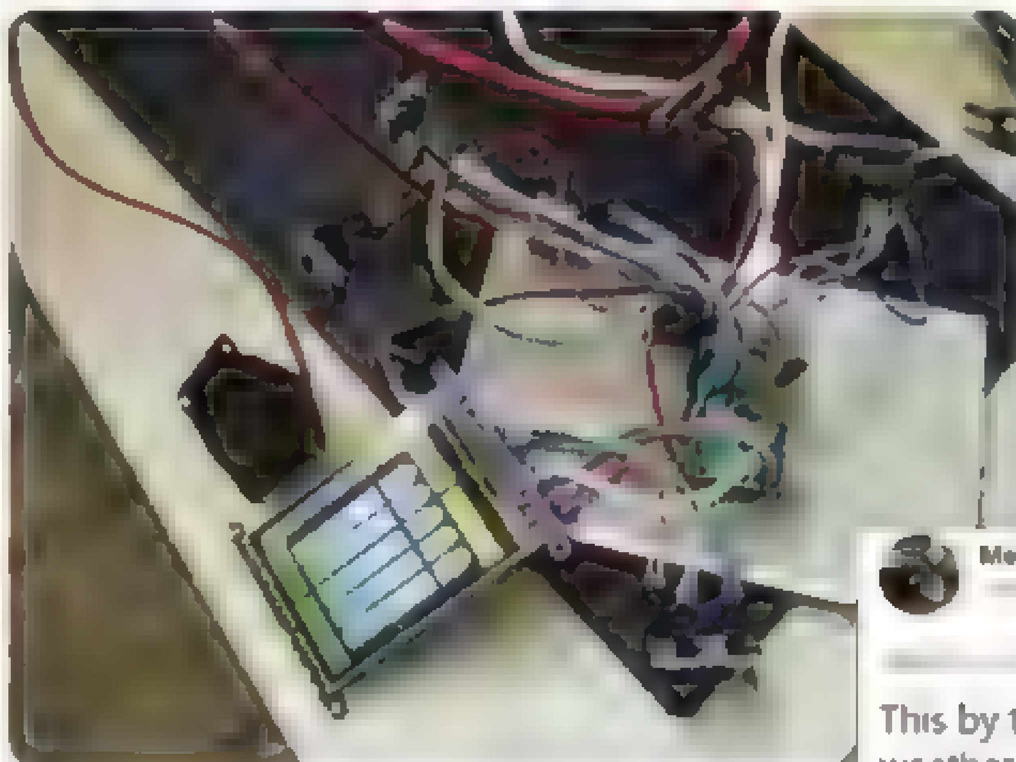
Here's a selection of some of the awesome things we got sent this month – and remember to follow along at the hashtag #MagPiMonday!

01. We like the little LCARS-inspired UI on the screen here
02. E-Ink panels always looks very clean. This UI really works well
03. Dr Footleg continues to make cool videos with his LED matrices
04. These sensors look very swish
05. We love these kinds of life simulators. They always look cool
06. A very fun concept, once again with e-Ink displays
07. The biscuit thief has been caught red-handed/jumpered
08. These pictures are stunning from RUHACam
09. This dithering method is spectacular!

01

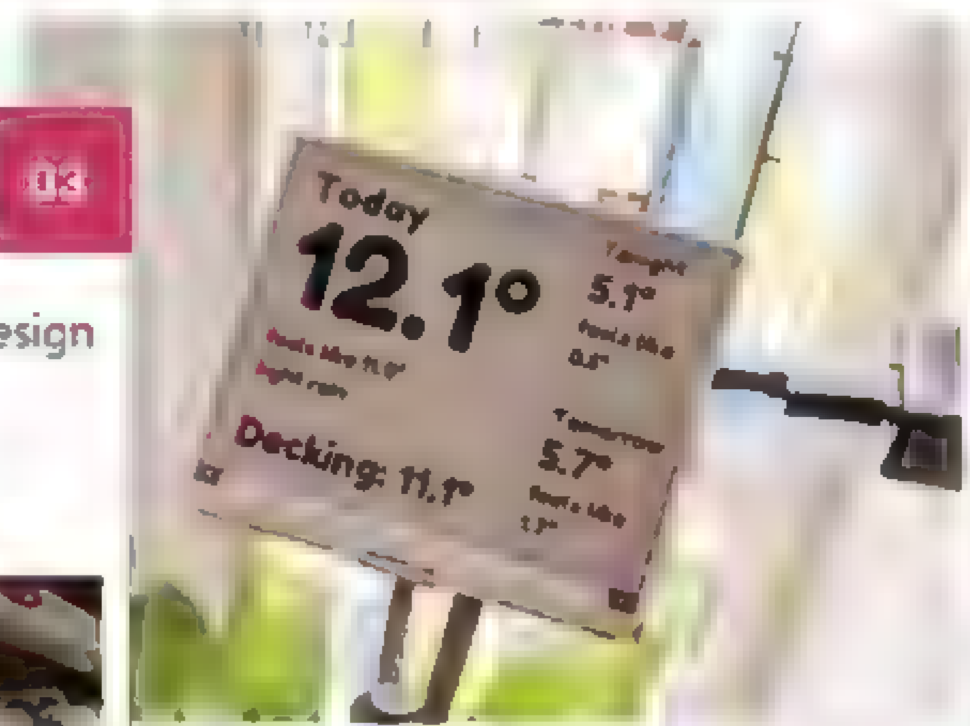
Michael (Mike) Horne

Gosh yes - epic week off and great weekend. Been working on my PicoPicorder project.
Full write-up of the Raspberry Pi Pico-based project here: recantha.co.uk/blog ...



02

This by the front door so I know if I need a coat. Area weather data comes from a JSON feed, and hyper-local readings from a sensor in the back garden on a Raspberry Pi Zero. All collated by another Raspberry Pi Zero in the house and output to an e-ink panel.



03

Dr Footleg - Roboteer
@drfootleg

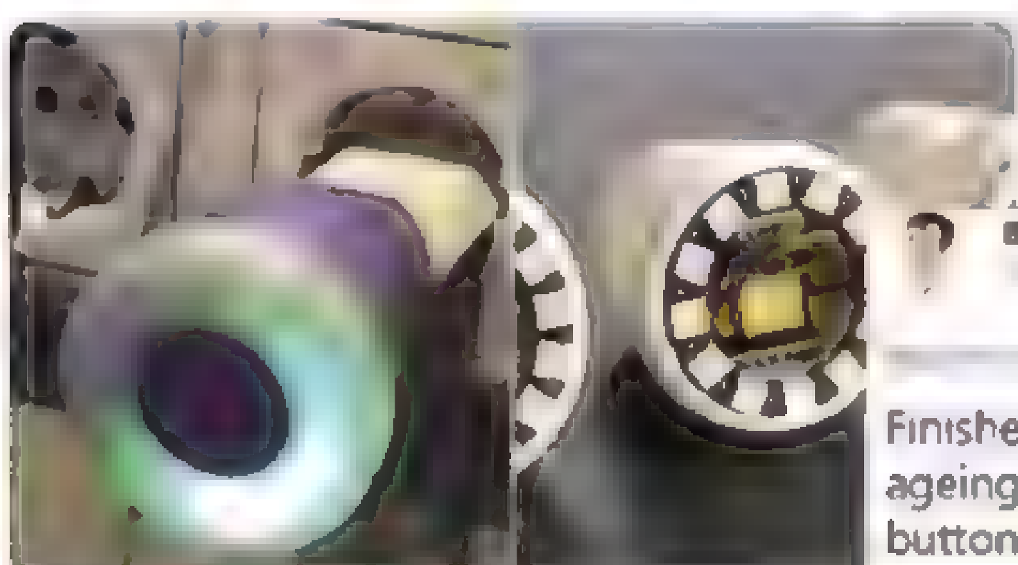
I soldered up the first PCB from my latest KiCAD design. An LED Matrix driver for the Pi. #MagPiMonda

04

TGD

I spent some time coding for PiMowBot project and Tim did an awesome job & designed new #3Dprint cases for a NeoPixel ring CO2 traffic light. Code ... New 3D models ...

#MINT #IOT #CO2Ampel #3dprint ... #esp8266 ... #RaspberryPi ...



05

Ben Hall

Finished off an interactive simulator of evolution and ageing! Mutant spread visualised on a panel of LEDs, buttons to introduce mutations. Runs on pi and pico!

06

Mike Fell

Here is v1 of my universal analog meter with eink display that can be set to any scale (here for example renewable elec on GB grid and moon phase). Parts ... Raspberry Pi zero, @waveshare00 display, ... gears. I'll put a video of it in action in the reply

07

Anne Carlill

This 'choc bics thief' guy is not getting the biscuit today. #raspberrypipico #MagPiMonday

08

Penk Chen
@penk

Replying to @TheMagPi

We open sourced the 3D printable retro-style Pi HQ camera, #RUHACam, and took it out for a spin in Tokyo

ruha camera

09

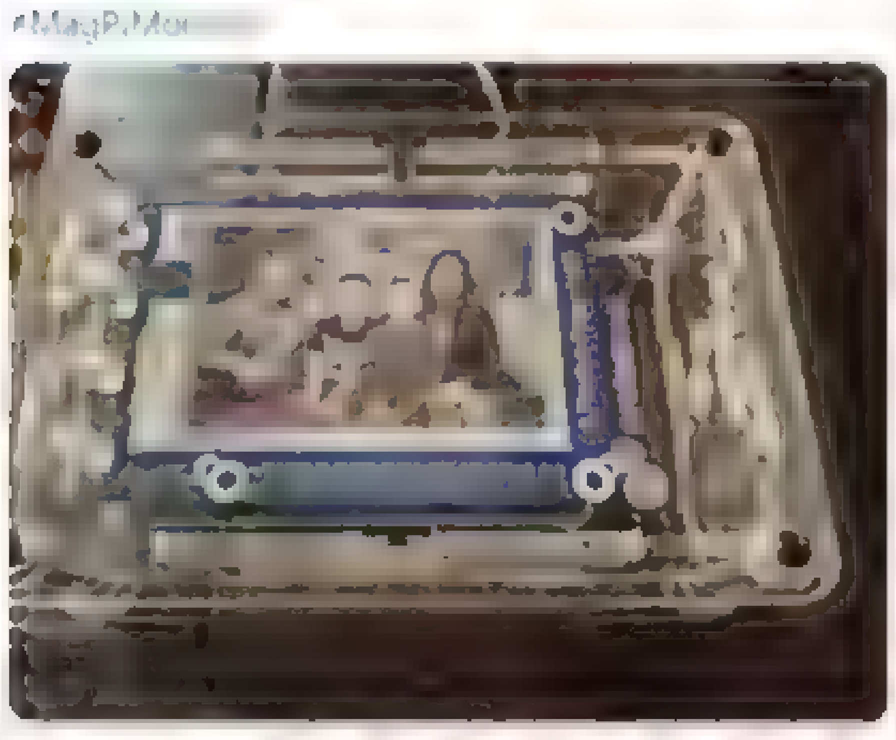
Sridhar Rajagopal

ePaper Photo Frame - writeup in progress.

It's impressive how much detail can be achieved with a tri-color display with an algorithm from 1976 -

And yes - that's a tri-color ePaper Display - Red, White and Black

#MagPiMonday



Crowdfund **this!**

Raspberry Pi projects you can crowdfund this month

USB RTC for Raspberry Pi

A simpler way to add a real-time clock to a Raspberry Pi – using USB! It has a low-power DS3231 RTC chipset and makes use of UART for connecting over USB. It's already been funded at the time of writing, so worth a look.

▶ **kck.st/39zGWDx**

LumiCube

"We think of the LumiCube as the ultimate Raspberry Pi kit. Perfect for beginners and enthusiasts alike. It takes the Raspberry Pi, a powerful mini computer, and combines it with a whole range of electronics and sensors, from a speaker to an accelerometer, all in a 10cm cube – the perfect playground for your creations."

▶ **kck.st/3lkBitv**

Pi Day donors

You helped out the Raspberry Pi Foundation with donations in March. Here's how it went

On 14 March, we celebrate Pi Day. In American notation, 14 March is 3/14, which is the same as the first three digits of the mathematical constant, pi. No emails about tau, please. To celebrate its namesake this year, the Raspberry Pi Foundation had a special fundraising campaign.

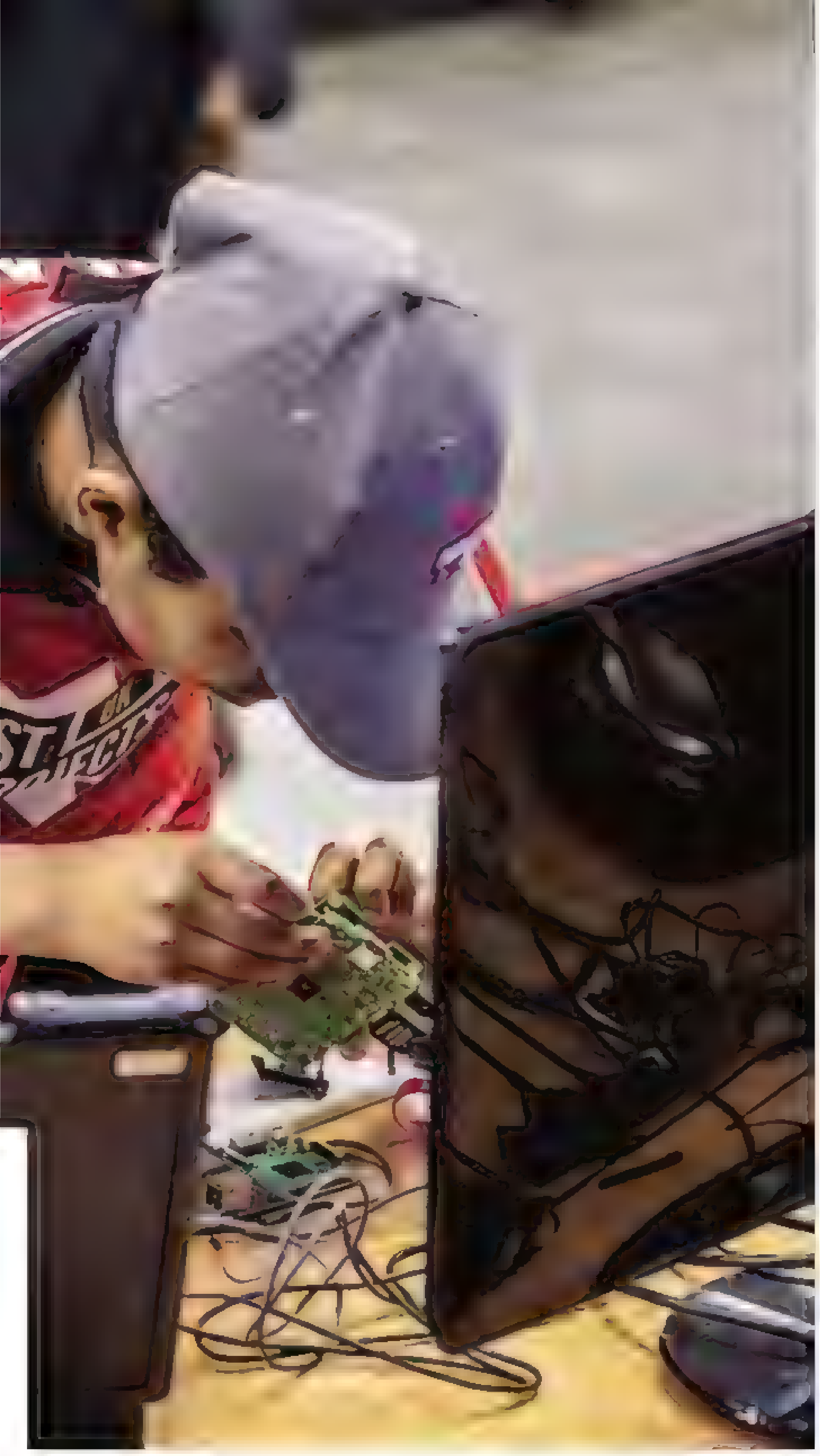
"When people think about Raspberry Pi, they usually think about the tiny, affordable, and powerful computers we all know and love," says Anna Coe, Senior Development Manager at the Raspberry Pi Foundation North America. "What many don't know is that the Raspberry Pi Foundation is a non-profit organisation with an educational mission. That mission is to put the power of computing and digital making into the hands of people all over the world."

There was a special little bonus for people who donated – their name mentioned here in *The MagPi*.

"In the weeks leading up to Pi Day, over 400 generous individuals and companies contributed more than £18,000 to support young learners who rely on the Raspberry Pi Foundation's free learning resources," Anna reveals. "Through gifts of $3.14, £31.42, €314.16, and more, Pi Day campaign donors have made an impact on 60,000 young digital makers who are learning new skills and getting creative with computing."

> Over 400 generous individuals and companies showed their support

We're honoured to display all your names. Thank you for funding the Raspberry Pi Foundation.



Special thanks from Raspberry Pi

The Raspberry Pi Foundation extends deep gratitude to the Pi Day campaign donors listed here, and to the additional 220 supporters who have chosen to remain anonymous. Special thanks goes to EPAM Systems, who matched the first $5000 raised dollar-for-dollar; to CanaKit for their generous gift of $3141; and to OKdo for donating 50% of their Pi Day weekend proceeds to this campaign.

If you would like to learn how you or your company can support the Raspberry Pi Foundation's mission, please visit **raspberrypi.org/support-us**.

Donors

COMPANIES

AAxios Technologies
CanaKit
Crosstalk Solutions
eduGOOGdroid.com
EPAM Systems
Hallsten Innovations Ltd
Logwood Computing Ltd
MINT Genie
OKdo
RaspberryTips
RICELEE
Tadintune Limited
Wild Computing Ltd
Woolsey Workshop

INDIVIDUALS

Alessandro Rani
Alex Penfold
Alexander Kirsch
Alister Ware
Alya Amarsy
Andrew Bennett
Andrew Book
Andrew Potter
Andrew Thomson
Andrew White
Andy Elder
Andy Felong
Antoine Vincent
Ariel Perez V.R.
Armand Christophe
Benjamin Howe
Bob Bailey
Brando Sabatini
Brian Coupe
Brian Fargo
Bruce Clawson
Carl Reasoner
Carl Smith
Carol Gonzales
Charles Godwin
Chloë Allen-Ede
Chris DeHut
Chris Kirk
Christian Bechheim
Christian Hiob
Christopher Bowring
Christopher Leadholm
Ciaran Byrne
Clayton Dymond
Clint Dimick
Damien Duddy
Daniel Dimmick
Daniel Harper
Daniel Pullan
David Ashworth
David Johnson
David Jones
David Rush
Dejun Yuan
Dirk Heinkelmann
Divyendu Singh
Dominic Varley
Dru Nelson
Ed Parsons
Efren Rodriguez
Eitan Sharon
Ella Sharon
Emma Staves
Ethan Banks
Everett Faircloth
Francisco de Assis
Barros de Menezes
Frank Dube
Garreth Tinsley
Garry Heather
Gary Thompson
Geoffrey Cross
Gerry Quinn
Gery Brosens
Gnanasekaran Thoppae
gojimmypi
Graham Fisher
Grahame Hambleton
Greg Parke
Grigori Fursin
Guy Leech
Harry Myhre
Hector Serrano
Hugh Cowan
Iván Oliva
Ivan Soldo
James Banks
James Copeland
James Duggan
James Pearson
James Turck
Jason Miller
Jason Townsend
Jay Roberts
Jeff Liu
Jeffrey Rollin
Jesper E. Siig
Joffrey Birster
Johann Blauensteiner
John Dallman
John Elliott
John Fitzpatrick
John Peart
Jonah Neeb
Jonathan
Jonathan Nourse
Jonathan Vannieuwkerke
Jorge Bailon
Jürgen Falch
Justin Driscoll
Justin Pinner
Justin Sauber
Kamilla Marosi
Kasper Holst
Keith Carscadden
Kevin McAleer
Kevin Taylor
Larry Howell
Laura Simms
Leah Yes
Lee Jordan
Len Layton
Leonard Wong
Linda Goetze
LK Ward
Luca Campisi
Lucas Dreher
Luis Alberto R. Antunez
Margot Thomas
Mark Routledge
Mark Seymour
Martin Woodward
Masafumi Ohta
Masami Mitsuhashi
Matt Heavner
Matt Sendorek
Matthew Sylvester
Michael Haß
Michal Krzywonos
Mohammad Asad
Murlidhhar Naidu
Neil Hoare
Nelson Hinman Jr
Nick Gushlow
Nick Kaufmann
Nick Sharp
Nick Stringer
Nick Twigg
Niklas Gertoft
Nithinut Ekapand
Patrick D.
Paul Clark
Paul Fretwell
Paul Gittere
Pavel Maly
Peter Becker
Peter Francis
Peter Taeleman
Peter Vincent
Phil Randal
Philip Ichinaga
Philip Mather
Poul Christiansen
R.C. Whiteley
Ralf Geschke
René Beckerschmidt
Ricardo de Azambuja
Rich Pearson
Richard Ash
Rob Sutton
Robert Sternberger
Robert Bradley
Rupert Wilson
Salvatore Del Pizzo
Samuel Pickard
Samyar Sadat Akhavi
Sarah Fawcett
Scott Bickley
Sean McManus
Shawn Bird
Shea Silverman
Simon Bartlett
Simon Reap
Simon Withers
Sinead Harold
Steffen Taube
Stephen Kellat
Stephen McGuinness
Steve Beck
Stewart Watkiss
Tero Hemiö
Thomas Proctor
Thomas Veach
Tihamer Benjamin Kovacs
Timothy Sailer
Tom Borg
Tom van den Enden
Trust your Imagination
Tyler Bramble
Venugopal Chidambaram
Walter Mollineaux
Yannick Bentz
Zachary Zebrowski

Your Letters

▲ *The Beginner's Guide* is packed full of all the info you need for getting started with Raspberry Pi

Binary to objects

I just got a Raspberry Pi. It's just like starting over for me; I like working in ones and zeroes and hexadecimal as I used to work on an Apple II.

Are there any good articles out there to help me convert from Apple II thinking to Raspberry Pi?

Donald via email

Unfortunately, we've not come across any one-to-one articles about going from Apple DOS to Raspberry Pi OS, or even coding in binary to Python. However, we think that *The Official Raspberry Pi Beginner's Guide* book (**magpi.cc/bguide4**) is a great way to start with Raspberry Pi, and should help you translate your skills.

Free GCSE lessons

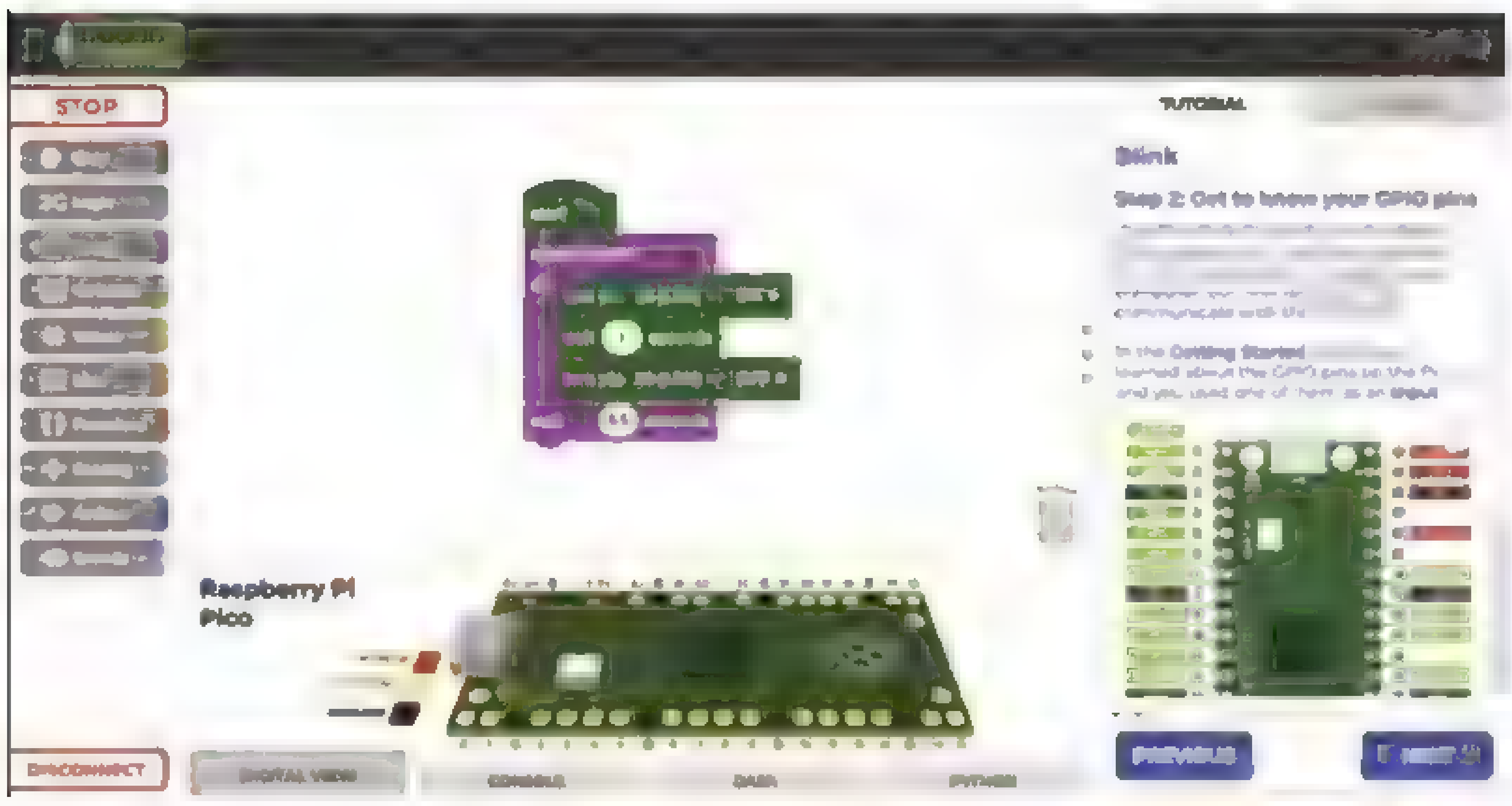
I noticed that the Isaac Computer Science website only currently does A levels. Could you recommend any other online learning sites, preferably free ones, that do a similar thing but with GCSEs? Are there any lesson plans on the projects website?

Kat via email

Good news: the Raspberry Pi Foundation has decided to expand the scope of the Isaac Computer Science platform to add GCSE content, thanks to a huge demand from teachers. Like the A level lessons, it will be available around the world, although tailored to the English curriculum. It should be launching early next year, and you can read more about it here: **magpi.cc/isaacgcse**.

▼ The current A level content is free and available to everyone

▼ The software looks a lot like Scratch and other block-based coding

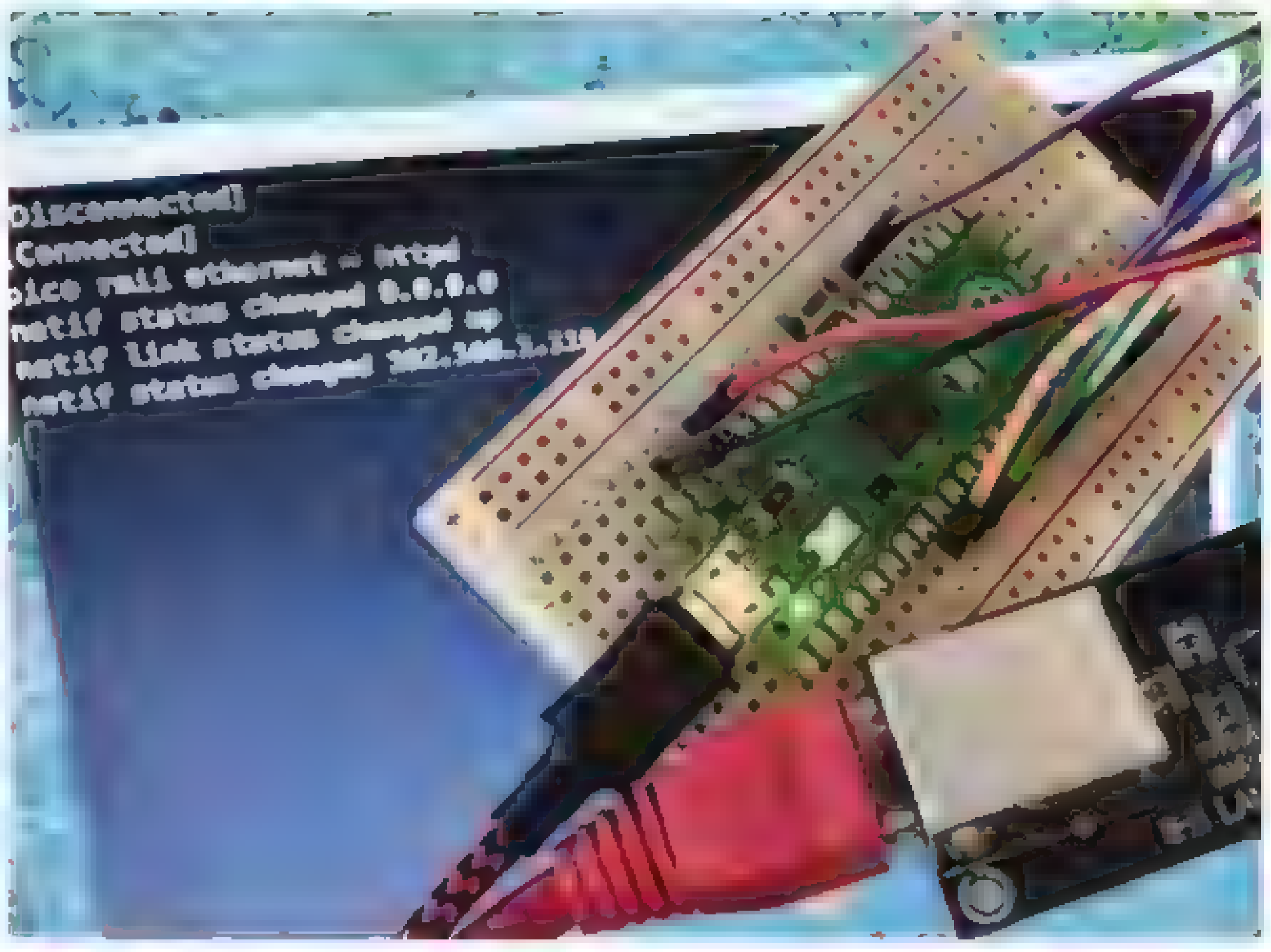


Scratch for Pico

I like using Scratch on Raspberry Pi to make programs. However, it doesn't seem to be available for Pico yet. Do you know if there will be a Scratch update soon that works with Pico? I think it would be really popular.

Ten via Facebook

While Scratch doesn't work with Raspberry Pi Pico yet, there is an alternative we can recommend called Piper Make. You can find it at **make.playpiper.com** and it's a free block-based programming language like Scratch. You can control circuits and sensors and such using the tools in Piper Make. And like other Piper stuff, it gives you challenges to complete, so you can learn and have fun at the same time.



▲ It requires a few wires and some special software, but it works

Pico networking

I got a Pico when it first came out, thanks to HackSpace magazine, and have been tinkering with it since, trying out examples from the books and such. One thing I'd like to do, though, is to add a wired connection to it. I've got a few gizmos to add WiFi, but there are certain places in my house that do not have a great wireless signal. Any ideas?

Jeegar via GitHub

You're in luck! While there's not a simple plug-and-play style solution, you can absolutely add an Ethernet port with some specific components, code, and wiring. Alasdair Allan, from Raspberry Pi, recently wrote up a blog on it, complete with a guide on how to get it working yourself. Check it out here: **magpi.cc/picoeth**.

Contact us!

- Twitter **@TheMagPi**
- Facebook **magpi.cc/facebook**
- Email **magpi@raspberrypi.com**
- Online **raspberrypi.org/forums**

WIN ONE OF TWO 4.01" ACEP 7-COLOUR E-PAPER DISPLAY HATS!

IN ASSOCIATION WITH The PiHut

Colour e-paper enables the creation of low-power, simple multicolour displays that look different and cool. This HAT has a 640×400 seven-colour display and fits neatly atop a full-size Raspberry Pi.

Head here to enter: magpi.cc/win | **Learn more:** magpi.cc/acepepaper

Terms & Conditions

Competition opens on **28 April 2021** and closes on **27 May 2021**. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.

Sizzling Summer PROJECTS

BLAZING HOT BUILDS FOR THE GREAT OUTDOORS

THE MAGPI #106 ON SALE 27 MAY

Plus!

Smart farming with Raspberry Pi

Put on your wearable tech

Install UNIX on Raspberry Pi Pico

DON'T MISS OUT!

magpi.cc/subscribe

TWITTER	@TheMagPi
FACEBOOK	fb.com/MagPiMagazine
EMAIL	magpi@raspberrypi.com

EDITORIAL

Editor
Lucy Hattersley
lucy@raspberrypi.com

Features Editor
Rob Zwetsloot
rob@raspberrypi.com

Sub Editor
Nicola King

ADVERTISING

Charlotte Milligan
charlotte.milligan@raspberrypi.com
+44 (0)7725 368887

DESIGN

criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Lucy Cowan, Sam Ribbits

Illustrator
Sam Alder

CONTRIBUTORS

Mike Cook, David Crookes, PJ Evans, Andrew Gillett, Gareth Halfacree, Martin O'Hanlon, Rosemary Hattersley, Nicola King, Phil King, KG Orphanides, Laura Sach, Mark Vanstone

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.com

Director of Communications
Liz Upton

CEO
Eben Upton

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6 The Enterprise Centre
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE
+44 (0)1293 312193
magpi.cc/subscribe
magpi@subscriptionhelpline.co.uk

This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine.

ISSN: 2051-9982.

A smart investment

Fantastic tales from the fourth dimension, featuring **Rob Zwetsloot**

Last month we had a fantastic feature from PJ Evans, a regular here at *The MagPi*, about home automation. As summer hurtles towards us with the momentum of a runaway freight train, you might be wondering if now is the right time to actually get stuck in with upgrading your home.

I'm here to tell you that now is the perfect time. As long as you have some time.

When we're out enjoying the parks and beaches and pub gardens, who will be left at home to feed the cat? Who will water your plants? Who will brew your artisanal latte? Certainly not the cat. So why not invest your time now so that you can have a bit more time when it really matters? Go find your copy of issue 104 of *The MagPi*, and have a look through the feature for some ideas and inspiration.

> As the saying goes, 'the time you enjoy wasting is not wasted time'

Wasted time

As the saying goes, 'the time you enjoy wasting is not wasted time'. And sometimes taking the extra time to set up a system to do a job feels like you're spending as much time as you might be saving. This is where we like to come in with fun ways to make with Raspberry Pi, and great stories to inspire you to dream big and maybe do something a bit different.

Maybe this is your excuse to get plants that need watering. Or to get into really nice coffee. Definitely not a cat, though: they're no fun. You might discover a fun new hobby, a new way to code, or at the very least, you might have a fun story to tell later on.

Nothing is forever

I have been there with the whole investing time for the future thing when it comes to making, so I do have some advice: build in some redundancies, make backups, write notes, and basically make sure that if something inevitably breaks, you can fix it. Of course, this may take more time, but it will save time in the future. Hopefully this won't happen while you're off enjoying your summer, though. Might overfeed the cat.

So go forth. Invest your time, have some fun, and then enjoy the sweet, sweet fruits of your labours.

AUTHOR

Rob Zwetsloot

Rob is amazing. He's also the Features Editor of *The MagPi*, a hobbyist maker, cosplayer, comic book writer, and extremely modest.

magpi.cc